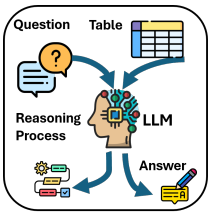
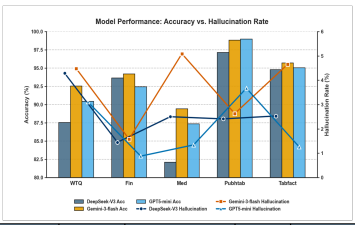
TrustTable: A Neuro-Symbolic Auditing Framework for Faithful Table QA

Guangzhen Zhao[†‡], **Dechang Kong**[†‡], **Tongyu Wu**[†‡], **Zhenjiang Dong**[†‡*]
† School of Computer Science, Nanjing University of Posts and Telecommunications, China
‡ The State Key Laboratory of Tibetan Intelligence, China
{zhaogz, 1025040707, 1225045346, dongzhenjiang}@njupt.edu.cn

Abstract

Large Language Models based Table Question Answering (LLMs-based TableQA) models excel in NLP field, however, they occasionally exhibit unfaithful behavior where correct answers are derived through erroneous reasoning paths. In this condition, we propose **TrustTable**, a neuro-symbolic framework designed to ensure reasoning faithfulness by auditing the reasoning processes of LLMs. Unlike monolithic LLM-based auditors, TrustTable decouples the auditing operation into two orthogonal dimensions. It enforces factual grounding by executing neurally generated Pandas code against the table, and ensures logical soundness by verifying reasoning chains through a LLM-synthesized formal solver. By integrating these symbolic checks, TrustTable enables a "Label-Free Audit Loop" that systematically identifies and rectifies reasoning flaws without human supervision. In addition, we present the TrustTable-Bench, a diagnostic dataset containing diverse error categories that range from calculation discrepancies to schema misalignments. This benchmark allows for a rigorous quantification of reasoning limitations. Experiments demonstrate that our symbolic audit detects reasoning flaws more accurately than advanced baselines. More broadly, TrustTable outperforms LLM judges in both majority voting with logical weighting and rejection sampling with process supervision.





Pos	No	Driver	Laps	Time/Retired	Grid	Points
1	9	Kimi Räikkönen	66	1:27:16.830	1	10
2	5	Fernando Alonso	66	+27.652	3	8
...	...	...	...	...	...	...
11	**12**	**Felipe Massa**	**63**	**Wheel rim**	**10**	
Ret	**11**	**Jacques Villeneuve**	**51**	**Engine**	**12**	
Ret	1	Michael Schumacher	46	Puncture	8	

Right answer &unfaithful reasoning
Question: Who was the last place driver to complete at least 50 laps?
Prediction: *Jacques Villeneuve*
Reasoning Trace:
The table lists drivers who completed at least 50 laps: Kimi Räikkönen (66 laps), Fernando Alonso (66),..., and Jacques Villeneuve (51 laps). *Jacques Villeneuve has the fewest laps among these (51), making him the last place driver to complete at least 50 laps.*

Wrong reasoning
Question: Which driver started the race from the 10th grid position?
Prediction: *Nick Heidfeld*
Reasoning Trace:
I looked at the **'Pos'** column in the table. *The row with 'Pos' equal to 10 contains the driver Nick Heidfeld.*

Figure 1: **Analysis of Reasoning Faithfulness in TableQA.** (Top) Metrics across datasets show that high answer accuracy often masks significant reasoning unfaithfulness. (Bottom) Representative examples compare the faithfulness gap (left), where models exploit spurious correlations to reach correct answers, with typical reasoning failures (right) that result in incorrect predictions. In the middle table, the `Grid` column denotes the *starting position* while the `Pos` column denotes the *finishing rank*; the model conflates the two, yielding a "right answer for wrong reason".

1 Introduction

The rise of Large Language Models (LLMs) has shifted Natural Language Processing from task-specific modules toward unified generative frameworks (Vaswani et al.; Wei et al., 2022a; Zhao et al., 2023). Consequently, Table Question Answering (TableQA) has become a central task for evaluating reasoning capabilities over semi-structured data. By leveraging Chain-of-Thought (CoT) (Wei et al., 2022b; Chen, 2023) prompting to elicit explicit reasoning paths, current state-of-the-art models have already surpassed human performance on several table-based benchmarks (Xie et al., 2022; Zhang et al., 2024).

However, high accuracy alone can be misleading. Standard TableQA evaluations primarily focus on the Exact Match (EM) of final answers, often overlooking the faithfulness of the reasoning process. The answer-centric approach masks a critical vulnerability: LLMs frequently generate reasoning

* Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 34907–34934

traces that serve as post-hoc rationalizations rather than the logical basis for their predictions (Turpin et al., 2023; Lanham et al., 2023). A concrete instance of the vulnerability is presented in Figure 1, drawn from the WikiTableQuestions benchmark (Pasupat and Liang, 2015). When asked to identify the "last place driver to complete at least 50 laps", the model correctly predicts "Jacques Villeneuve". However, its Chain-of-Thought (CoT) reveals a spurious correlation. The model justifies the answer by claiming that Villeneuve had the "fewest laps" (51), which incorrectly equates a lower lap count with a lower finishing position. While the flawed heuristic happens to yield the correct entity here, it lacks robustness and fails when lap counts do not align with rank. Our preliminary analysis identifies a non-trivial incidence of reasoning hallucinations, affecting approximately 5% of correct predictions (see hallucination rate, Figure 1, top-right). In these cases, models often misinterpret semantic relationships between columns or execute flawed arithmetic that fortuitously aligns with the ground truth. Such silent failure modes render these systems unreliable for critical applications in finance (Chen et al., 2021) and healthcare (Akhtar et al., 2022), where auditability is paramount.

To ensure such reliability, it is imperative to verify the reasoning trajectory, not just the final output. While process-based supervision has flourished in closed-world mathematical domains (e.g., Process Reward Models (Lightman et al., 2024; Uesato et al., 2022)), transplanting this success to TableQA remains non-trivial. This challenge stems from two fundamental barriers: **First**, TableQA imposes a strict dual requirement of logical validity and *faithful grounding*. Unlike pure math, tabular reasoning must navigate semi-structured data where intermediate steps demand precise anchoring to specific table cells. Current verifiers frequently overlook "spurious correctness", failing to detect instances where correct final answers are derived from hallucinated data values. **Second**, the heterogeneity of table operations (e.g., filtering, aggregation) necessitates a verification mechanism capable of simultaneous symbolic and semantic auditing. While recent works have introduced fine-grained atomic skill annotations (Zhang et al., 2025), they fundamentally rely on *neural verifiers*. Such neural-centric approaches primarily assess whether a step *semantically aligns* with a skill definition but struggle to enforce *rigorous grounded checks*. Consequently, they fail to intercept deceptive reasoning flaws where the generated text appears linguistically fluent but remains logically invalid under symbolic scrutiny. This gap highlights the urgent need for a unified, automated framework tailored for the neuro-symbolic verification of table reasoning.

To bridge this gap, we propose TrustTable, a **model-agnostic Neuro-Symbolic Chain-of-Thought Verification Framework**. We advocate for a paradigm shift from "neural semantic evaluation", which relies on the opaque intuition of LLMs to "white-box symbolic auditing". Specifically, TrustTable reconceptualizes the neural CoT not as free text, but as a latent symbolic program. Unlike previous monolithic verifiers, our framework orthogonally decomposes the verification task into two specialized streams: a Pandas-based *FactChecker* that enforces strict data grounding, and a Z3-based *LogicAuditor* (de Moura and Bjørner, 2008) that validates logical satisfiability. By imposing these rigorous execution constraints, TrustTable establishes a "Generate-Audit-Refine" loop, ensuring that no answer is accepted unless its reasoning path is mathematically provable and factually anchored. We will release our code and datasets at `https://github.com/chrischowfy/TrustTable`. The main contributions are summarized as follows:

1. We propose TrustTable, a neuro-symbolic framework that shifts verification from "neural prediction" to "symbolic auditing". By decomposing reasoning traces into executable data operations and logical constraints, our framework ensures that verification is deterministic and grounded.

2. We design a closed-loop mechanism integrating hybrid solvers with label-free refinement. Utilizing a *Pandas FactChecker* for grounding and a *Z3 LogicAuditor* for satisfiability, the system intercepts "Right Answer & Wrong Reason" errors and triggers iterative self-correction using symbolic feedback.

3. We release TrustTable-Bench, a diagnostic dataset and evaluation protocol for auditing reasoning faithfulness. Leveraging this testbed alongside standard benchmarks, we demonstrate that TrustTable achieves State-of-the-Art performance, effectively mitigating the "Right Answer & Wrong Reason" trap.

2 Related Work

2.1 Table Reasoning with LLMs

The landscape of Table QA has evolved from parametric approximation to deterministic execution. Initial methodologies focused on augmenting table comprehension via large-scale fine-tuning (Gong et al., 2020; Liu et al., 2022) or eliciting intermediate reasoning via Chain-of-Thought (CoT) prompting (Wei et al., 2022b; Kojima et al., 2022). To bridge the dissonance between linear reasoning and tabular topology, recent frameworks like *Chain-of-Table* (Wang et al., 2024) and *Table-as-Thought* (Sun et al., 2025) introduced dynamic schema manipulation. However, these parametric approaches remain fundamentally constrained by stochastic hallucinations (Liu et al., 2024) and function as *open-loop systems*, where erroneous structural transformations propagate irreversibly without external validation.

To enforce deterministic rigor, Neuro-Symbolic approaches have emerged as a robust alternative by translating reasoning into executable programs. Frameworks such as Binder (Cheng et al., 2023) and Program-of-Thoughts (Chen et al., 2023) utilize interpreters to offload numerical computation, while recent hybrid paradigms like TabSQLify (Nahid and Rafiei, 2024) and H-STAR (Abhyankar et al., 2025) decompose queries into executable symbolic representations. Despite these advancements, existing methods predominantly adopt a linear "Neural-to-Executor" paradigm that treats code generation merely as a functional *solver*. They process any syntactically correct code without evaluating the semantic plausibility of the underlying logic. Our work addresses this gap by introducing a white-box auditing paradigm, employing symbolic engines not just to compute answers, but as an intermediate constraint layer to rigidly enforce data grounding and logical satisfiability.

2.2 Reasoning Faithfulness and Process Verification

Amidst concerns regarding the faithfulness of LLMs, verifying the validity of CoT has emerged as a critical imperative. Early strategies like Self-Consistency (Wang et al., 2023) attempted to filter errors by aggregating answers via majority voting. However, the model may generate the same erroneous logic (e.g., recurring calculation errors), rendering the majority vote factually incorrect. To enforce more rigorous checking, recent frameworks have turned to formal logic and granular decomposition (Quan et al., 2024; Feng et al., 2025). VeriCoT (Feng et al., 2025) exemplifies the logical approach by auto-formalizing CoT steps into First-Order Logic for solver-based verification. However, such methods face a "Semantic-Symbolic Gap" in tabular contexts: they verify *logical validity* (internal consistency) rather than *computational correctness*, often failing to ground premises in factual table data. Similarly, decomposition-based methods like Atomic Reasoning (Zhang et al., 2025) break verification into modular atomic skills to reduce cognitive load. Although this improves generation quality, it remains a generative enhancement rather than a true auditing mechanism, as the execution of each atomic skill still relies on the probabilistic and potentially ungrounded output of the neural model.

Distinguishing our work from these paradigms, we propose a label-free Neuro-Symbolic Auditing framework. Unlike the former focus on logical entailment or Atomic Reasoning's reliance on neural execution, our framework employs deterministic symbolic engines as external auditors.

3 Construction of TrustTable-Bench

To rigorously evaluate reasoning faithfulness, we introduce TrustTable-Bench, constructed under a Three-Layer Decoupling protocol to guarantee experimental integrity and preclude circular self-validation. Specifically, (L1) External seed provenance: all seeds are drawn exclusively from publicly available community benchmarks; (L2) Independent adversarial generation: reasoning chains are produced by general-purpose LLMs (GPT-5) under strict informational isolation from our verifier's Pandas/Z3 constraints; (L3) Human-verified gold-standard annotation: final labels are assigned by independent annotators with high inter-annotator agreement. This protocol proceeds in three phases:

① **Seed Curation via Task Unification.** We aggregate diverse contexts from financial (SciAutomic (Zhang et al., 2025)), open-domain (WTQ (Pasupat and Liang, 2015)), and medicine sources (PubHealthTab (Akhtar et al., 2022), SciAutomic (Zhang et al., 2025)). Notably, for verification-based datasets, we apply a Claim-to-Question Transformation to unify declarative claims into a standard QA format $(Q, \mathcal{T}, A)$.

② Four-Quadrant Diagnostic Generation. Employing a Counterfactual Reverse-Generation strategy, we synthesize reasoning chains across four topological quadrants to test specific failure modes: (i) Type 1: Faithful ($Z^+ \wedge A^+$): Valid logic (Z^+) yielding correct answers (A^+). (ii) Type 2: Spurious ($Z^- \wedge A^+$): Hallucinated logic (Z^-) accidentally yielding correct answers. (iii) Type 3: Fallacious ($Z^- \wedge A^-$): Logical errors leading to incorrect answers (A^-). (iv) Type 4: Inconsistent ($Z^+ \wedge A^-$): Valid logic with perturbed final execution.

③ Quality Assurance. A human-in-the-loop validation on a stratified sample confirms a 96.5% validity rate with high inter-annotator agreement (Fleiss' $\kappa = 0.88$) (Fleiss, 1971). See Appendix §B for full generation details.

4 Methodology

4.1 Problem Formulation

Formally, let $\mathcal{T}$ be the table context and Q be the question. We treat the reasoning chain as a Latent Symbolic Program π, synthesized to satisfy both grounding ($\mathcal{V}_{\text{ground}}$) and logical ($\mathcal{V}_{\text{logic}}$) constraints. The validity function $\mathcal{V}$ is defined as $\mathcal{V}(\pi, \mathcal{T}) \triangleq \mathcal{V}_{\text{ground}}(\pi, \mathcal{T}) \wedge \mathcal{V}_{\text{logic}}(\pi, \mathcal{T})$. Our inference objective seeks a program π^* that maximizes the likelihood under a neural generator P_θ while strictly satisfying $\mathcal{V}$:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, P_\theta(\pi \mid Q, \mathcal{T}) \quad \text{s.t.} \quad \mathcal{V}(\pi, \mathcal{T}) = \textsc{Pass} \quad (1)$$

This constrained optimization is intractable to solve directly and is thus approximated via an iterative refinement process. Upon finding a valid π^*, the final answer is obtained deterministically as $A = \text{Exec}(\pi^*, \mathcal{T})$.

4.2 Overall Framework

Figure 2 illustrates TrustTable, a model-agnostic auditing framework operating on a candidate tuple $(Q, \mathcal{T}, \hat{y})$. The pipeline begins with a Neural Decomposer, which dissects the raw CoT $\hat{y}$ into atomic steps and projects them into dual symbolic representations. These are scrutinized by a Hybrid Symbolic Verifier via a tripartite protocol: (1) The *FactChecker* executes Pandas scripts to validate data grounding, intercepting Type 3 (Hallucination) errors; (2) The *LogicAuditor* employs a Z3 Solver to enforce logical axioms, detecting Type 2 (Spurious Logic) via proof by contradiction; and (3) A *Consistency Monitor* verifies the alignment between execution results and textual claims (Type 4). Finally, an Iterative Refinement Mechanism consolidates verification failures into an Audit Report, triggering a "Label-Free Audit Loop" that guides the *Refiner* ($\mathcal{M}_{\text{repair}}$) to rectify either the verification tooling or the reasoning chain itself (detailed in §4.5), without ground truth supervision.

4.3 Semantic Decomposition and Program Synthesis

To transition from unstructured neural reasoning to deterministic symbolic auditing, we employ a two-stage transformation pipeline: Atomic Decomposition followed by Dual-Path Symbolic Projection.

Atomic Decomposition. Given a raw reasoning trace Z (generated by any off-the-shelf LLM), our first objective is to discretize the continuous text into a sequence of verifiable units. We define a parsing function $f_{\text{parse}} : Z \to \Pi$, which decomposes the trace into atomic steps $\Pi = \{s_1, s_2, \ldots, s_n\}$. As implemented in our pipeline, a LLM-based parser classifies each step s_i into one of two modes based on its semantic function: 1) **Fact Steps (τ = fact):** Operations involving data retrieval, entity extraction, or row filtering (e.g., "Locate the row for *Brazil*"). 2) **Inference Steps (τ = inf):** Operations involving logical deduction, arithmetic calculation, or comparative reasoning (e.g., "Since 19 > 10, Brazil wins").

Dual-Path Symbolic Projection. Decomposition alone does not eliminate the ambiguity inherent in raw text. To enable rigorous auditing, we therefore transform these atomic units into executable symbolic domains using a Program Synthesizer $\mathcal{M}_{\text{code}}$. This module generates dual representations for parallel verification:

- **For Fact Steps:** The synthesizer maps the natural language claim to a Pandas Verification Function $f_{\text{v}}(df) \to \{0, 1\}$. This script encapsulates the grounding logic (e.g., `df['Country'].str.contains('Brazil')`) to be executed against the table context.
- **For Inference Steps:** The synthesizer translates the reasoning into Z3 Constraints (SMT-LIB format). This involves formalizing the premises and the conclusion into logical assertions suitable for satisfiability checking.

This projection effectively bridges the gap between the *probabilistic* nature of the LLM's thought pro-

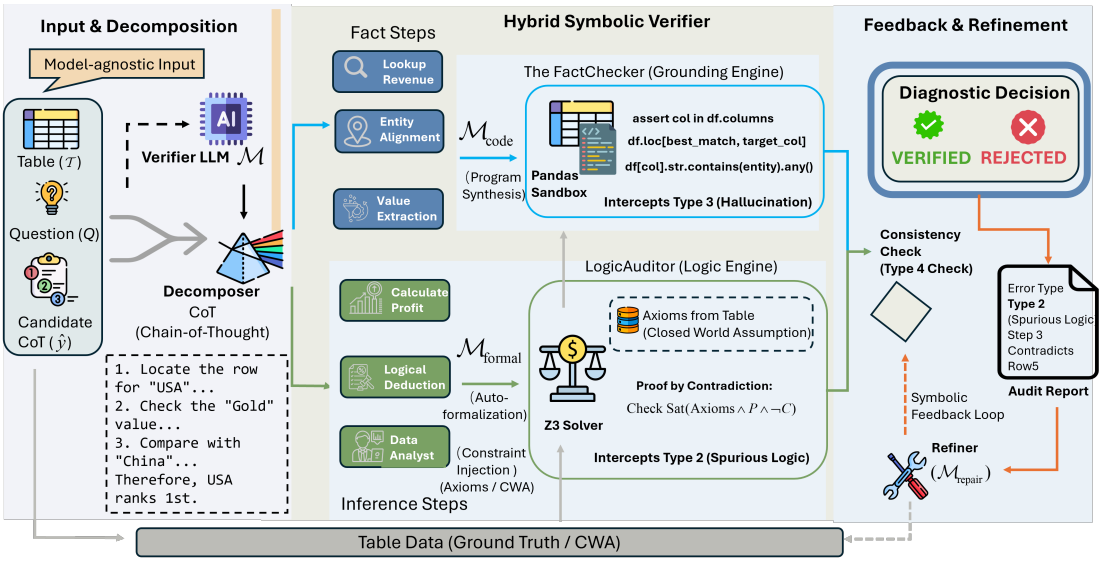


Figure 2: The TrustTable Framework. A closed-loop neuro-symbolic auditing system comprising: (1) Atomic Decomposition of reasoning traces; (2) Dual-Path Verification using Pandas (Grounding) and Z3 (Logic); and (3) Iterative Refinement for self-correction without reliance on gold labels.

cess and the *deterministic* requirements of the symbolic verifiers.

4.4 Hybrid Symbolic Verifier

The core of TrustTable is the *Hybrid Symbolic Verifier*, a deterministic engine that executes the synthesized programs to enforce rigorous consistency across the diagnostic quadrants defined in Appendix §B.

① The FactChecker: Executing Grounding Logic. For steps projected as Pandas functions, the *FactChecker* executes them directly against the raw table dataframe $\mathcal{T}$. Critically, this module intercepts Type 3 (Ungrounded) errors: if the function returns `False` or encounters an execution error (e.g., `KeyError: 'Col_X'`), it signals a Grounding Violation, indicating that the model is citing non-existent data.

② The LogicAuditor: Proving Satisfiability. For steps projected as Z3 constraints, we rigorously check logical soundness by framing verification as a Proof by Contradiction. The solver operates within a constructed constraint environment defined by three layers:

(i) Data Grounding Constraints ($\mathcal{C}_{data}$): We serialize relevant table rows into the Z3 context with Distinctness Constraints, ensuring a Closed World Assumption (CWA).

(ii) Type-Safety Constraints ($\mathcal{C}_{type}$): Numerical values are mapped to `z3.Real` and entities to `z3.String`, strictly prohibiting invalid cross-type operations.

(iii) Entailment Verification ($\mathcal{C}_{logic}$): We assert the axioms of the table ($\mathcal{A}_{\mathcal{T}}$), the premises ($P$), and the negation of the conclusion (C).

$$\text{Validity} = \neg\text{SAT}(\mathcal{A}_{\mathcal{T}} \wedge P \wedge \neg C) \quad (2)$$

If the Solver (based on the axioms in Appendix §C) returns UNSAT, the step is logically entailed. If it returns SAT, the solver has found a counterexample, flagging the step as a Spurious Correlation. This rigorously intercepts Type 2 (Right Answer & Wrong Reason) errors that outcome-based metrics fail to detect.

③ Consistency Monitor. To prevent Execution Inconsistency (Type 4), we perform a final alignment check. We compare the result of the deterministically executed program sequence A_{exec} with the LLM's textual answer A_{text}. Any discrepancy ($A_{\text{exec}} \neq A_{\text{text}}$) triggers an Inconsistent Signal, forcing the model to resolve the conflict between its derived reasoning and stated conclusion.

4.5 Label-Free Iterative Refinement

Standard verification methods typically function as passive filters that discard invalid samples, wasting computational effort. TrustTable advances this paradigm by establishing a constructive Label-Free Audit Loop, in which the Refiner $\mathcal{M}_{\text{repair}}$ plays two complementary roles depending on the failure source:

① **Tool Repair.** When the synthesized Pandas or Z3 verification program itself fails to execute (e.g., `KeyError` on a mismatched column, malformed SMT-LIB declarations), the feedback message $\mathcal{F}_{\text{tool}}$ carries the runtime traceback, and $\mathcal{M}_{\text{repair}}$ regenerates the verification script so that auditing can proceed. This stage ensures that any subsequent REJECT signal reflects a genuine reasoning flaw rather than a tooling artifact.

② **CoT Repair.** When the verifiers execute cleanly and emit a substantive REJECT (Grounding Violation, Spurious Logic, or Inconsistency), the feedback message $\mathcal{F}_{\text{cot}}$ encodes the specific reasoning failure (e.g., *"Z3 Solver found contradiction: derived value 15 violates constraint..."*), and $\mathcal{M}_{\text{repair}}$ rewrites the reasoning chain $\hat{y}$.

We model both forms of refinement as conditional generation:

$$\pi^{(t+1)} \sim P_\theta(\pi \mid Q, \mathcal{T}, \pi^{(t)}, \mathcal{F}) \tag{3}$$

where $\mathcal{F} \in \{\mathcal{F}_{\text{tool}}, \mathcal{F}_{\text{cot}}\}$ depending on the failure source. The entire loop is capped at $K = 3$ iterations and is independent of gold answers, strictly adhering to realistic inference settings where ground truth is unavailable.

4.6 Theoretical Analysis: Error Bound Reduction

To rigorously substantiate TrustTable, we analyze the inference process through an Information Theoretic lens (full derivation in Appendix §E). Standard E2E models approximate discrete symbolic operations via continuous representations, suffering from intrinsic *approximation noise*. We term this irreducible residual entropy as Computational Friction (ϵ_{comp}), forcing a theoretical lower bound on the E2E error: $H_{\text{E2E}}(Y|\mathcal{X}) \geq H^*(\mathcal{Y}|\mathcal{X}) + \epsilon_{\text{comp}}$.

In contrast, TrustTable achieves structural convergence via two pathways. First, by delegating execution to a deterministic engine, the conditional execution entropy vanishes ($H(Y|Z) \equiv 0$), mathematically guaranteeing the elimination of ϵ_{comp}. Second, we enforce Orthogonal Disentanglement of constraints: *Factual Grounding* (Valuation Space via Pandas) and *Logical Satisfiability* (Function Space via Z3). Due to the statistical independence of these error modes, the Mutual Information gain is additive, maximizing the reduction of reasoning uncertainty:

$$H_{\text{Ours}}(Y|\mathcal{X}) \to H^*(\mathcal{Y}|\mathcal{X}) + \underbrace{H_{\text{reason}}(Z|\mathcal{X})}_{\text{Prior Uncertainty}} - \underbrace{\sum_{k\in\{pd,z3\}} I(Z;\mathcal{V}_k)}_{\text{Additive Information Gain}} \tag{4}$$

This confirms that TrustTable achieves a strictly lower theoretical error bound than monolithic approaches by structurally eliminating friction and maximizing verification coverage.

5 Experiments

5.1 Experimental Setup

Dataset. We evaluate our framework on TrustTable-Bench, the diagnostic dataset constructed in Section B. It contains 12,000 samples evenly distributed across four reasoning categories ($Z^{\pm}A^{\pm}$), covering Finance, Medical & Public Health, and Open domain.

Evaluation Metrics. Let $v_i \in \{\text{ACC}, \text{REJ}\}$ denote the verification decision. We define four core metrics: (1) Verified Correct Answer Rate (VCAR) measures strict faithfulness by penalizing ungrounded correct answers: $\text{VCAR} = \frac{1}{|\{x\in\mathcal{D}|x\in\text{Type 1}\}|}\sum\mathbb{I}(v_i = \text{ACC} \wedge x_i \in \text{Type 1})$. (2) Diagnostic Interception Rate (DIR) evaluates defense against specific traps (Type 2/4) via the rejection recall: $\text{DIR}_k = \frac{\sum_{x\in\mathcal{D}_k}\mathbb{I}(v=\text{REJ})}{|\mathcal{D}_k|}$. (3) Faithfulness Precision (FP) quantifies the purity of accepted samples: $\text{FP} = P(x \in \text{Type 1}|v = \text{ACC})$. Please refer to **Appendix §A** for the formal mathematical definitions and calculation details of these metrics.

Baselines. We evaluate TrustTable against four representative paradigms: 1) **Standard CoT** (Wei et al., 2022b): The *Parametric Reasoning* baseline, eliciting step-by-step natural language reasoning without external tools. 2) **Program-of-Thoughts (PoT)** (Chen et al., 2023): The *Program-Aided* baseline, which decouples computation by synthesizing executable Python code for numerical subtasks. 3) **Atomic Skills** (Zhang et al., 2025) This method decomposes complex claims into atomic steps and verifies each step sequentially. 4) **Veri-CoT** (Feng et al., 2025): A *Logic-based Verification* method that translates natural language reasoning into First-Order Logic for symbolic consistency checking.

Implementation Details We implement TrustTable using a modular neuro-symbolic stack. We

Table 1: Diagnostic Performance on TrustTable-Bench. **Bold** / underline: panel best / second-best within each column.

Method	VCAR	DIR (Safety)		FP
	(Faith.)	Spur.	Inc.	(Trust)
Panel A: Finance (FinQA) – Logic Heavy				
PoT (Chen et al., 2023)	82.5	85.2	96.4	70.9
Atomic Skills (Zhang et al., 2025)	26.1	91.5	95.7	59.4
VeriCoT (Feng et al., 2025)	78.0	31.3	28.2	18.3
Standard CoT (Wei et al., 2022b)	**86.6**	87.2	91.0	63.2
TrustTable (Ours)	78.5	**94.8**	**98.8**	**84.7**
Panel B: Medical & Public Health (MedQA + PubHealthTab) – Fact Heavy				
PoT (Chen et al., 2023)	67.0	90.2	92.8	68.2
Atomic Skills (Zhang et al., 2025)	56.6	92.6	91.6	70.4
VeriCoT (Feng et al., 2025)	26.3	90.3	**96.8**	57.2
Standard CoT (Wei et al., 2022b)	**84.2**	88.7	83.9	64.2
TrustTable (Ours)	83.7	**94.3**	96.0	**81.3**
Panel C: Open Domain (WTQ) – Fuzzy Mixed				
PoT (Chen et al., 2023)	76.0	84.3	96.2	63.3
Atomic Skills (Zhang et al., 2025)	48.5	91.7	90.2	60.2
VeriCoT (Feng et al., 2025)	56.0	92.3	94.7	75.8
Standard CoT (Wei et al., 2022b)	76.5	83.8	65.5	45.2
TrustTable (Ours)	**76.6**	**92.9**	**97.2**	**82.6**

employ DeepSeek-V3 as the backbone, utilizing a dual-temperature strategy: we set $T = 0.7$ for the *Reasoning Generator* to encourage hypothesis diversity, while freezing $T = 0.0$ for the *Program Synthesizer* to ensure deterministic code generation. During inference, the *Label-Free Iterative Refinement* loop is capped at a maximum depth of $K = 3$.

5.2 Experimental Results

5.2.1 Diagnostic Performance on TrustTable-Bench

As shown in Table 1, the five methods occupy distinct operating points determined by their verification mechanism. Following the panel structure of prior work, we report results over three diagnostic panels: Finance (FinQA), Medical & Public Health (MedQA + PubHealthTab merged), and Open Domain (WTQ). While neural baselines such as Standard CoT achieve high VCAR in logic-heavy domains (e.g., 86.6% on Finance), their overall reliability is compromised by opaque acceptance criteria that admit spurious reasoning. In contrast, TrustTable achieves consistent VCAR across all three panels (78.5 / 83.7 / 76.6%), confirming that our dual-constraint mechanism (Schema ∩ Logic) preserves faithful reasoning without erroneously rejecting valid complex operations. The modest VCAR concession on Finance (8.1 pt below Standard CoT) reflects strict Pandas/Z3 grounding of numeric claims that a neural verifier would accept on confidence alone.

The structural advantage of neuro-symbolic decoupling is most pronounced in error interception. *(1) Defense against Spurious Logic:* program-based methods (PoT) exhibit a critical auditing void, with DIR_{spur} falling to 85.2% on Finance and 84.3% on Open Domain. This failure stems from PoT's reliance on execution validity rather than semantic grounding. TrustTable enforces strict grounding via the *FactChecker*, achieving the highest DIR_{spur} on every panel (94.8 / 94.3 / 92.9%) and effectively pruning "Right Answer & Wrong Reason" hallucinations. *(2) Defense against Inconsistency:* while Standard CoT performs adequately on Finance (91.0%) and poorly on Open Domain (65.5%), TrustTable's *Consistency Monitor* attains 98.8, 96.0, and 97.2% interception across the three panels, second only to VeriCoT on Medical (96.8%). This rigor translates directly into trustworthiness: TrustTable yields the highest FP across all domains, averaging 82.9% with a peak of 84.7% on Finance, a lead of +13.8 pt over the strongest baseline (PoT 70.9%). FP functions as a reliability proxy, indicating that accepted reasoning from TrustTable is substantially more likely to be both physically grounded and logically sound than that from any neural baseline.

Inference: Logic-Weighted Voting. Standard Self-Consistency (SC) converges slowly and, as k grows, *systematic biases* in the policy accumulate and can dominate the vote, causing accuracy to peak early and mildly decline rather than plateau. We address both issues with a *Logic-Weighted Voting* strategy that assigns weight $w = 0$ to paths rejected by TrustTable.

As shown in Figure 3, this *symbolic pruning* yields a higher accuracy plateau and the flattest long-range profile. Standard SC reaches 82.7% at $k = 5$ and 83.2% at $k = 10$, peaks at 83.5% near $k \approx 16$, and drifts back to 82.8% by $k = 30$; adding a PoT verifier delays the drift to $k \approx 17$ (peak 84.0%) but does not prevent it, since it catches execution errors but not spurious grounding. TrustTable reaches $\mathbf{84.5}\%$ at $k = 8$ and stays within a 0.2-point band through $k = 30$, because the Pandas/Z3 verifier prunes spurious paths before they accumulate votes. *Verified quality* thus lets the system lock onto the optimal solution faster and *sustain* it as the budget grows.

Training: Alignment via Symbolic Distillation. Beyond inference auditing, TrustTable acts as a rigorous filter for model alignment. We employ *Process-Supervised Rejection Sampling Fine-*

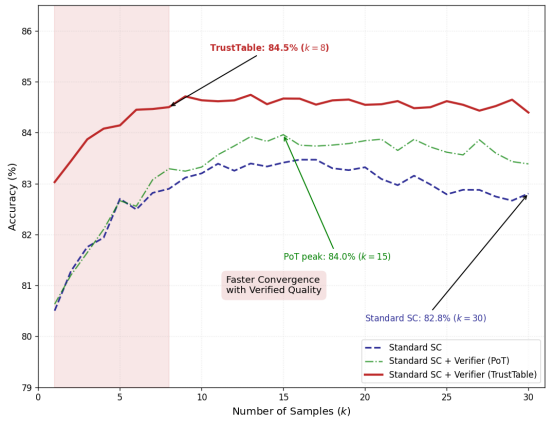


Figure 3: TrustTable converges to its optimal performance range significantly faster (k = 8) than the Standard SC baseline, which requires k = 16+ samples to stabilize.

Tuning (RFT) to distill symbolic rigor from a teacher (DeepSeek-V3) into a student (Qwen2.5-7B). Standard outcome-based supervision retains any reasoning path yielding the correct answer, inevitably polluting the training set with spurious correlations (Type 2 errors). In contrast, our Symbolic Distillation filters generated trajectories via the TrustTable auditor, retaining only those that pass strict Grounding, Logic, and Consistency checks. As shown in Table 2, training on this purified subset ($\mathcal{D}_{\text{process}}$, 72% of original) is far more effective than the full outcome-based dataset. The student model achieves a +3.4% accuracy gain and a dramatic reduction in Spurious Acceptance Rate (SAR) from 12.5% to 5.2%, demonstrating that the model effectively internalizes symbolic constraints.

Table 2: Comparison of Qwen2.5-7B fine-tuned on Outcome-filtered vs. TrustTable-verified data (WTQ).

Strategy	Data Size	Acc	SAR ↓	FP ↑
Outcome-based	100%	64.7	12.5	63.2
TrustTable (Ours)	**72%**	**68.1**	**5.2**	**81.4**

5.2.2 Case Study: The "Right Answer & Wrong Fact" Trap

To demonstrate the necessity of semantic verification, we present a case from the WikiTableQuestions diagnostic subset (item `nu-11`; DeepSeek-chat generation). This example illustrates a subtle **Grounding Hallucination (Type 2)**, where the CoT fabricates an intermediate numeric value ("John's Total is 20") but the final entity-level answer ("John") still coincidentally matches the gold label.

Table 3: Ablation Study. We dissect module contributions across two dimensions: (1) **Detection**: Impact on Spurious Logic Detection (DIR_{spur}); and (2) **Refinement**: Impact on Correction Success Rate (CSR).

Configuration	Detection (DIR_{spur})		Refinement
	WTQ	FinQA	CSR (Avg)
Panel A: Constraint Modules			
Full Framework	**92.9%**	**94.8%**	**48.4%**
w/o Grounding†	87.4% ↓	85.1%	–
w/o Logic Check	90.2%	85.9% ↓	–
Panel B: Feedback Mechanism			
w/o Sym. Feedback *(Blind Retry)*	–	–	34.1% ↓

PoT's verdict is semantically wrong despite the string-level match: the CoT's reasoning path is unfaithful — it fabricates an intermediate numeric value ("John's Total is 20") that does not exist in the table. Because PoT independently re-computes the final answer rather than auditing the reasoning, it is structurally blind to this class of error. TrustTable's atomic FactChecker operates *at the level of individual reasoning steps*: every quantity the CoT cites (here, "20") is compared against the actual cell value (12), and the mismatch is reported regardless of whether the downstream conclusion happens to be correct. This case concretely demonstrates the "Right Answer & Wrong Fact" trap discussed in §1 and justifies our *process-oriented* verification over answer-oriented baselines.

5.2.3 Ablation and Component Analysis

We perform a component-wise ablation study to dissect the individual contributions of the error detection and self-correction modules, as summarized in Table 3. Panel A evaluates Spurious Logic Detection (DIR_{spur}) under systematic module removal. Removing the Pandas-based FactChecker drops DIR_{spur} on WTQ from 92.9% to 87.4% (-5.5 pt), showing that entity and cell-level grounding is the dominant lever on fuzzy open-domain tables. Disabling the Z3 logic check causes a much larger drop on the calculation-intensive FinQA panel, where DIR_{spur} falls from 94.8% to 85.9% (-8.9 pt). The two modules are complementary rather than redundant: each owns the error type its symbolic backend is designed to catch, and removing either leaves a substantial gap on the matched domain.

Panel B evaluates the system's resilience via the *Correction Success Rate (CSR)*, defined as the fraction of initially rejected reasoning chains that are

Table 4: **Comparison on a Grounding Hallucination Case (WTQ nu-11).**

Component	PoT (Baseline)	TrustTable (Ours)
Question	"Does Pat or John have the highest total?"	
Gold answer	John	
CoT (from generator)	*"Pat Baldwin has a Total of 1. When I check John O'Flynn's stats, I see he has a Total of **20**, which is the highest number in that column."*	
Actual table values	Pat Baldwin Total = 1; John O'Flynn Total = **12**	
Claimed final answer	John (correct)	
Auditor mechanism	re-computes `solve(df)` returns `"john"`	per-step FactChecker code: `df.loc[df["Name"]=="John O'Flynn", "Total"]`
Auditor evidence	`pot_answer='john' == claimed='John'`	`actual='12' != claimed='20'` at CoT Step 2
Decision	**ACCEPT** (string match)	**REJECT** (grounding mismatch)

successfully revised into valid, grounded traces. The full Label-Free Refiner reaches 48.4% CSR, while the "Blind Retry" variant that resamples without structured error messages reaches only 34.1% (-14.3 pt). This gap reflects the value of *deterministic symbolic feedback*: instead of a generic "try again" prompt, TrustTable injects localized diagnostics (e.g., "Value Mismatch: 19 vs 5" or "KeyError on column `Pos`"), reframing the task from open-ended regeneration into a *constrained rewrite*.

5.3 Efficiency Analysis

Cost-Effectiveness on WikiTableQuestions. To assess the practical viability of our framework, we evaluate the trade-off between performance and total inference budget on the WikiTableQuestions (WTQ) test set, using *Average API Calls* per query as a hardware-agnostic efficiency proxy that accounts for both generation and step-wise verification overhead. Unlike Self-Consistency (SC), which enforces a static brute-force sampling budget, TrustTable employs a *dynamic budget* strategy: straightforward queries are resolved in a single pass, while iterative refinement is triggered only for the subset of samples rejected by our verifier.

As shown in Table 5, this mechanism remains economical even after accounting for verification cost. TrustTable improves accuracy over the Greedy CoT baseline from 80.1% to 83.4% ($+3.3$ pt) at an average cost of 5.80 calls per query. To reach comparable accuracy, SC requires a blind budget of k=10 (83.2%, $1.7\times$ the call volume), and SC at k=5 remains 0.7 pt behind TrustTable at a similar budget (82.7% vs 83.4% at 5.00 vs 5.80 calls). TrustTable thus matches or exceeds the strongest SC configuration while reducing the total API volume by 42.0% relative to k=10, indicating that *verified quality* is a more efficient driver of performance than *unverified quantity*.

Verifier Reliability: Human Alignment. The validity of this dynamic budget hinges on the verifier's precision—avoiding unnecessary refinements for correct answers while intercepting errors. To validate this, we engaged three experts to annotate 200 reasoning chains. As shown in Table 6, our neuro-symbolic solver achieves Agreement ($\kappa = 0.86$) with human judgments, significantly outperforming purely neural verifiers (LLM Self-Check). This high fidelity ensures that computational resources are invested only where necessary.

Table 5: Comparison of Accuracy and Total API Calls.

Method	Acc	Avg Calls	Call Reduction
Greedy CoT	80.1%	1.00	Baseline
SC ($k = 5$)	82.7%	5.00	+400% Cost
SC ($k = 10$)	83.2%	10.00	+900% Cost
TrustTable (Ours)	**83.4%**	**5.80**	**-42.0%** (vs k=10)

Table 6: Alignment between Verifier decisions and Human Experts.

Verifier Type	Agreement (%)	Cohen's κ
LLM Self-Check (DeepSeek-V3)	88.6%	0.74
VeriCoT (Logic-only)	91.0%	0.79
TrustTable (Hybrid)	**93.5%**	**0.86**

6 Conclusion

We presented **TrustTable**, a neuro-symbolic auditing framework that ensures tabular faithfulness by scrutinizing the *validity of reasoning processes* rather than outcome probabilities. Through atomic decomposition and dual-path auditing, our approach intercepts reasoning process hallucinations. Empirically, TrustTable enhances both inference efficiency and model alignment, where our Dense Process Rewards guide student models to internalize constraints via Rejection Sampling Fine-Tuning. Ultimately, we hope this work paves the way for further research into process-oriented supervision, enabling more faithful and self-correcting systems in high-stakes environments.

Limitations

While TrustTable establishes a rigorous standard for tabular processing auditing, we acknowledge two primary limitations:

1. Scope of Logical Reasoning. Our framework is currently optimized for deterministic reasoning tasks (e.g., arithmetic calculation, strict ranking, and entity grounding) where logic can be unambiguously mapped to hard Z3 constraints. It is not yet adapted for probabilistic or "soft" logic, such as the trend analysis or vague qualitative comparisons often found in scientific verification datasets (e.g., SCITAB). Translating these soft linguistic nuances into rigid symbolic representations remains an open challenge for future neuro-symbolic research.

2. Structural Complexity. TrustTable operates on flattened 2D DataFrames. Consequently, it may experience performance degradation when processing hierarchical tables with complex headers or merged cells (common in raw PDF documents). The current linearization process risks losing structural topology in such cases. Future iterations could integrate structure-aware encoders (e.g., TAPAS) to enhance grounding robustness on complex table formats.

Acknowledgments

The authors thank the anonymous reviewers for their valuable comments. This work is supported by the Jiangsu Provincial Key Research and Development Program (No. BE2023025), and the Scientific Research Fund of Nanjing University of Posts and Telecommunications (No. XK0040924146 and No. XK0040925024).

References

Nikhil Abhyankar, Vivek Gupta, Dan Roth, and Chandan K Reddy. 2025. H-star: Llm-driven hybrid sql-text adaptive reasoning on tables. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2025)*, pages 8841–8863, Albuquerque, New Mexico.

Mubashara Akhtar, Oana Cocarascu, and Elena Simperl. 2022. Pubhealthtab: A public health table-based dataset for evidence-based fact checking. In *Findings of the Association for Computational Linguistics: NAACL 2022 (Findings of NAACL 2022)*, pages 1–16, Seattle, United States.

Wenhu Chen. 2023. Large language models are few (1)-shot table reasoners. In *Findings of the association for computational linguistics: EACL 2023 (Findings of EACL 2023)*, pages 1120–1130, Dubrovnik, Croatia.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. FinQA: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021)*, pages 3697–3711, Online and Punta Cana, Dominican Republic.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. 2023. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*, Kigali, Rwanda.

Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory*, 2nd edition. John Wiley & Sons, Hoboken, NJ.

Leonardo de Moura and Nikolaj Bjørner. 2008. Z3: An efficient SMT solver. In *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008)*, pages 337–340, Budapest, Hungary.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Sean Welleck, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and fate: limits of transformers on compositionality. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS 2023)*, pages 70293–70332, New Orleans, LA, USA.

Yu Feng, Nathaniel Weir, Kaj Bostrom, Sam Bayless, Darion Cassel, Sapana Chaudhary, Benjamin Kiesl-Reiter, and Huzefa Rangwala. 2025. Vericot: Neuro-symbolic chain-of-thought validation via logical consistency checks. *arXiv preprint arXiv:2511.04662*.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Heng Gong, Yawei Sun, Xiaocheng Feng, Bing Qin, Wei Bi, Xiaojiang Liu, and Ting Liu. 2020. TableGPT: Few-shot table-to-text generation with table structure reconstruction and content matching. In *Proceedings of the 28th International Conference on*

Computational Linguistics (COLING 2020), pages 1978–1988, Barcelona, Spain (Online).

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS 2022)*, New Orleans, LA, USA.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukošiūtė, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, and 11 others. 2023. Measuring faithfulness in chain-of-thought reasoning. *Preprint*, arXiv:2307.13702.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*, Vienna, Austria.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. TAPEX: Table pre-training via learning a neural SQL executor. In *International Conference on Learning Representations (ICLR 2022)*, Virtual.

Tianyang Liu, Fei Wang, and Muhao Chen. 2024. Rethinking tabular data understanding with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2024)*, pages 450–482, Mexico City, Mexico.

William Merrill and Ashish Sabharwal. 2024. The expressive power of transformers with chain of thought. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*, Vienna, Austria.

Md Mahadi Hasan Nahid and Davood Rafiei. 2024. TabSQLify: Enhancing reasoning capabilities of LLMs through table decomposition. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2024)*, pages 5725–5737, Mexico City, Mexico.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL 2015)*, pages 1470–1480, Beijing, China.

Xin Quan, Marco Valentino, Louise Dennis, and Andre Freitas. 2024. Enhancing ethical explanations of large language models through iterative symbolic refinement. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2024)*, pages 1–22.

Zhenjie Sun, Naihao Deng, Haofei Yu, and Jiaxuan You. 2025. Tables as thought: Exploring structured thoughts in LLM reasoning. In *Proceedings of the 4th Table Representation Learning Workshop*, pages 19–33, Vienna, Austria.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS 2023)*, pages 74952–74965, New Orleans, LA, USA.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Advances in Neural Information Processing Systems (NIPS 2017)*, pages 5998–6008, Long Beach, CA, USA.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations (ICLR 2023)*, Kigali, Rwanda.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. In *The Twelfth International Conference on Learning Representations (ICLR 2024)*, Vienna, Austria.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS 2022)*, pages 24824–24837, New Orleans, LA, USA.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle,

Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, and 4 others. 2022. UnifiedSKG: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022)*, pages 602–631, Abu Dhabi, United Arab Emirates.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024. Tablellama: Towards open large generalist models for tables. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2024)*, pages 6024–6044.

Yuji Zhang, Qingyun Wang, Cheng Qian, Jiateng Liu, Chenkai Sun, Denghui Zhang, Tarek Abdelzaher, Chengxiang Zhai, Preslav Nakov, and Heng Ji. 2025. Atomic reasoning for scientific table claim verification. *arXiv preprint arXiv:2506.06972*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2023. A survey of large language models. *Preprint*, arXiv:2303.18223.

A Evaluation Metrics Details

In this section, we provide rigorous mathematical formulations for the metrics used in our evaluation, establishing a direct correspondence between our diagnostic quadrants and system performance.

A.1 Formal Definitions

Let $\mathcal{D}$ be the diagnostic test set. For each sample x_i, the system yields a verification decision $v_i \in \{\text{ACCEPT}, \text{REJECT}\}$ and a final output pair $(\hat{z}_i, \hat{y}_i)$. We categorize the output based on the ground truth topology: Type 1 (Z^+A^+), Type 2 (Z^-A^+), Type 3 (Z^-A^-), and Type 4 (Z^+A^-).

1. Verified Correct Answer Rate (VCAR). VCAR serves as our strict "Faithfulness Recall" metric. Unlike standard accuracy, it penalizes correct answers derived from incorrect logic (Type 2), counting only rigorously verified correct samples:

$$\text{VCAR} = \frac{\sum_{x \in \mathcal{D}} \mathbb{I}(v_i = \text{ACCEPT} \wedge x_i \in \text{Type 1})}{|\{x \in \mathcal{D} \mid x \in \text{Type 1}\}|} \tag{5}$$

2. Diagnostic Interception Rate (DIR). To evaluate the specific defense capability against different reasoning pitfalls, we define DIR as the recall of the verifier on specific error subsets $\mathcal{D}_k$:

$$\text{DIR}_k = \frac{\sum_{x \in \mathcal{D}_k} \mathbb{I}(v = \text{REJECT})}{|\mathcal{D}_k|} \tag{6}$$

We report DIR_{spur} for k = Type 2 (evaluating Grounding/Logic Verifiers) and DIR_{inc} for k = Type 4 (evaluating the Consistency Monitor).

3. Faithfulness Precision (FP). FP evaluates the *trustworthiness* of the system's acceptance. It represents the conditional probability that an accepted sample is genuinely faithful, treating Type 2, 3, and 4 as "leaked" risks:

$$\text{FP} = \frac{N_{\text{Type 1}}}{N_{\text{accepted}}} = \frac{\sum_{x \in \mathcal{D}} \mathbb{I}(v = \text{ACCEPT} \wedge x \in \text{Type 1})}{\sum_{x \in \mathcal{D}} \mathbb{I}(v = \text{ACCEPT})} \tag{7}$$

A.2 Metric-to-Data Mapping

Table 7 elucidates the alignment between evaluation metrics and the reasoning quadrants.

B Construction of TrustTable-Bench

Existing benchmarks typically provide only ground truth answers, lacking annotations for the correctness of the intermediate reasoning process. To rigorously evaluate our framework's faithfulness across diverse domains, we constructed **TrustTable-Bench**, a diagnostic dataset comprising 12,000 samples comprehensively covering four reasoning quadrants. The construction pipeline consists of three phases: Seed Data Curation, Counterfactual Reverse-Generation, and Human Validation.

B.1 Phase 1: Seed Data Curation

We selected four representative datasets to serve as Seed Contexts, ensuring broad coverage across Finance, Public Health, Medicine, and Open Domain. As shown in Table 8, we processed these heterogeneous sources into a unified Question-Answering format $(Q, \mathcal{T}, A_{\text{gold}})$ to serve as the ground truth basis for generation.

Financial Question Answering. We curated the ATOMIC_Fin subset from the ATOMIC dataset (Zhang et al., 2025), specifically by reformulating original declarative claims into interrogative questions. Derived from S&P 500 earnings reports, this subset features high information density and requires rigorous answer derivation. Over 40% of the questions involve multi-step numerical reasoning (e.g., "What is the ROI ratio..."), serving as a primary testbed for complex table QA tasks.

Public Health & Medicine. We incorporated PUBHEALTHTAB (Akhtar et al., 2022) and the Medical subset of ATOMIC. PUBHEALTHTAB tests robustness against noisy, non-standard tables. To adapt these "Fact Verification" datasets (Claim → Label) for Question Answering, we employed a *reverse generation method* using DeepSeek-V3, rewriting declarative claims into questions where the key entities serve as the answers.

General Logic (WIKITABLEQUESTIONS). WTQ (Pasupat and Liang, 2015) provides operational diversity, including aggregation (sum, average), superlatives, and sorting. It prevents the model from overfitting to domain-specific linguistic styles.

B.2 Phase 2: Four-Quadrant Diagnostic Generation

Using the unified seed data, we employed SOTA Large Language Models (e.g., GPT5) as data generators. We utilized a Counterfactual Reverse-Generation strategy to synthesize reasoning chains. For each seed example, we generated 6 distinct

Table 7: **Mapping of Evaluation Metrics to Diagnostic Quadrants.** Symbols: Z^+ (Valid Logic), Z^- (Invalid Logic), A^+ (Correct Answer), A^- (Incorrect Answer).

Metric	Target Subset	Ideal Action	Objective & Interpretation
I. Faithfulness (Recall of Truth)			
VCAR (Verified Correct Answer Rate)	**Type 1: Faithful** ($Z^+ \wedge A^+$)	**ACCEPT**	**Metric:** Retention Rate of Z^+A^+. *Goal:* Ensure valid reasoning is not falsely rejected.
II. Safety (Defense against Traps)			
DIR - Spurious (DIR_{spur})	**Type 2: Spurious** ($Z^- \wedge A^+$)	**REJECT**	**Metric:** Rejection Rate of Z^-A^+. *Goal:* Intercept "Right Answer, Wrong Reason" (Hallucinations).
DIR - Inconsistent (DIR_{inc})	**Type 4: Inconsistent** ($Z^+ \wedge A^-$)	**REJECT**	**Metric:** Rejection Rate of Z^+A^-. *Goal:* Detect execution mismatches or code generation errors.
III. Trustworthiness (Purity)			
FP (Faithfulness Precision)	**Global Accepted** (All Types)	**MAXIMIZE** **Type 1 Ratio**	**Metric:** Precision of Acceptance. *Goal:* Ensure that if the user sees an answer, it is verified.

Table 8: Statistics of the Seed Datasets used to construct TrustTable-Bench.

Seed Source	Domain	Tables	Examples	Processing
SCIATOMIC	Finance	247	500	Claim → Question
WTQ	Open Domain	500	500	Direct Usage
PUBHEALTHTAB	Public Health	463	500	Claim → Question
SCIATOMIC	Medical	382	500	Claim → Question
Total	-	**1,592**	**2,000**	-

variants to ensure granular coverage of error types (Total $N = 2,000 \times 6 = 12,000$):

1. Faithful Reasoning & Correct Answer ($Z^+ \wedge A^+$). Generated: 1 sample/seed. This subset represents the "Gold Standard" ($\mathcal{D}_{Gold}$). We prompted the generator to derive step-by-step reasoning strictly grounded in $\mathcal{T}$ to reach the ground truth A_{gt}. These samples test the framework's Recall, i.e., the verifier must accept valid logic.

2. Spurious Reasoning & Correct Answer ($Z^- \wedge A^+$). Generated: 3 samples/seed. This subset targets Spurious Correlations (Type 2). To evaluate fine-grained diagnostic capabilities, we systematically synthesized three distinct error modes via targeted adversarial prompting:

- **Grounding Error (Z^-_{ground}):** The model reaches the correct answer but cites irrelevant cells or non-existent columns.
- **Calculation Error (Z^-_{calc}):** The model derives the correct number through erroneous arithmetic (e.g., "10 + 10 = 30" → hallucinate "30").
- **Logic Error (Z^-_{logic}):** The model uses a flawed reasoning path (e.g., 'MAX' instead of 'MIN') that coincidentally aligns with the ground truth.

These samples serve as "Traps" to test the framework's **Precision**.

3. Fallacious Reasoning & Incorrect Answer ($Z^- \wedge A^-$). Generated: 1 sample/seed. This subset represents standard reasoning errors ($\mathcal{D}_{Error}$). We injected logical noise into the chain, leading to incorrect answers. These samples evaluate basic error detection capability.

4. Valid Reasoning & Mismatched Answer ($Z^+ \wedge A^-$). Generated: 1 sample/seed. This subset targets Execution Inconsistency (Type 4). We took valid Type 1 samples and artificially perturbed the final textual answer while keeping the correct reasoning chain intact. These samples test the framework's Self-Consistency Check.

B.3 Phase 3: Quality Assurance

To ensure label rigor, we implemented a comprehensive human-in-the-loop validation protocol involving four NLP graduate students. All annotators were fully informed about the purpose of the study, the nature of the dataset (public benchmarks), and how their annotations would be used for system evaluation prior to their participation. They were compensated via standard research assistantship stipends at university rates. Spanning four weeks (avg. 2h/day), the annotators independently reviewed a stratified random sample of 10% of the full dataset ($N = 1,200$). For each triplet $(Q, \mathcal{T}, CoT)$, annotators performed a blind re-annotation, classifying the generation quality into five mutually exclusive categories without access to the intended labels:

1. **Accurate & Faithful (Z^+A^+):** Correct answer, reasoning strictly follows table evidence.

2. **Accurate & Unfaithful (Z^-A^+):** Correct answer, flawed reasoning (sub-classified into *Grounding*, *Calculation*, or *Logic* errors).
3. **Incorrect & Unfaithful (Z^-A^-):** Both answer and logic are flawed.
4. **Incorrect & Faithful (Z^+A^-):** Valid logic, wrong execution result due to perturbations.
5. **Not Evaluable:** Unintelligible output.

Human Validation Protocol. A generated sample is deemed *valid* if and only if the human-annotated category aligns with the intended generation target. For instance, if the pipeline aimed to synthesize a *Spurious* sample but the annotator classified it as *Faithful* (indicating a failure to follow adversarial instructions), the sample is marked as invalid. As detailed in Table 9, this rigorous evaluation yielded an average data validity rate of 96.5% and a high inter-annotator agreement of Fleiss' $\kappa = 0.88$.

Table 9: Quality statistics of TrustTable-Bench ($N = 1,200$). *Validity* denotes the percentage of samples confirmed by humans to match their topological definitions.

Category Topology	Count (10%)	Validity (%)	Agreement (κ)
Type 1: Faithful (Z^+A^+)	200	98.0	0.92
Type 2: Spurious (Z^-A^+)	600	94.5	0.85
- Grounding Error	*(200)*	*95.0*	*0.86*
- Calculation Error	*(200)*	*94.0*	*0.84*
- Logic Error	*(200)*	*94.5*	*0.85*
Type 3: Fallacious (Z^-A^-)	200	97.5	0.90
Type 4: Inconsistent (Z^+A^-)	200	96.0	0.86
Overall	**1,200**	**96.5**	**0.88**

Model-Based Difficulty and Diagnostic Value. To verify the adversarial value of TrustTable-Bench, we sampled 200 instances from each category and evaluated representative SOTA backbones: DeepSeek-V3, Gemini-3-Flash, and GPT-5-mini. As shown in Table 10, the results reveal a structural **"Verification Gap"** that none of the prompting strategies fully closes:

1. **Trade-off between Type 1 acceptance and Type 4 rejection.** Standard CoT achieves higher Type 1 acceptance (82.4–85.7%) by trusting the model's own reasoning, but pays for it with weaker Type 4 detection (80.1–83.4%). Program-of-Thoughts (PoT) inverts this trade: by re-executing claims as code, it boosts Type 4 rejection to 95.1–96.8% but drops Type 1 acceptance by roughly 7 pt (75.2–79.5%) due to false rejections of valid but unconventional reasoning. No single prompting strategy is uniformly best on the deterministic axes alone.

Table 10: **Dataset Difficulty Assessment.** Evaluation of SOTA models (Standard CoT vs. PoT) on TrustTable-Bench. The high rejection rate on Type 4 confirms models possess strong execution checking capabilities, whereas the performance drop on Type 2 highlights the specific challenge of semantic grounding.

Model Backbone	Type 1 (Faithful)	Safety: Rejection Rate (↑)		Trust
	Acc (↑)	Type 2 (Spurious)	Type 4 (Inconsistent)	FP (↑)
Prompt Strategy: Standard CoT				
DeepSeek-V3	82.4%	86.6%	80.1%	57.5%
Gemini-3-flash	84.0%	84.3%	81.5%	55.2%
GPT-5-mini	85.7%	88.5%	83.4%	60.8%
Prompt Strategy: Program-of-Thoughts (PoT)				
DeepSeek-V3 (Code)	75.2%	86.6%	95.1%	67.5%
Gemini-3-flash (Code)	77.8%	84.1%	96.0%	65.3%
GPT-5-mini (Code)	79.5%	88.9%	96.8%	70.2%

2. **Type 2 detection saturates near 85% across all configurations.** Both Standard CoT (84.3–88.5%) and PoT (84.1–88.9%) plateau in the same narrow band on spurious-reasoning rejection, regardless of backbone. This convergence shows that detecting "Right Answer, Wrong Reason" hallucinations is not a capability that scales with model strength alone, nor is it solved by re-executing computations: the missing ingredient is grounding the cited evidence against the actual table cells, which neither neural nor code-based verification provides.

3. **FP remains below 71% even for the strongest backbone.** Faithfulness Precision peaks at 70.2% (GPT-5-mini + PoT), meaning that nearly one in three accepted answers is still backed by unfaithful reasoning. This persistent ceiling across both prompt strategies and three backbones confirms the diagnostic value of TrustTable-Bench: the benchmark exposes a verification bottleneck that cannot be closed by upgrading the underlying language model. The remaining gap is precisely what symbolic auditing in TrustTable is designed to address.

C Solver Constraint Axioms

To rigorously validate a reasoning step z_t given the tabular context $\mathcal{T}$ and the history of verified premises $\mathcal{H}_{t-1}$, our Z3Auditor constructs a symbolic constraint system Φ. The verification is framed as a Satisfiability Modulo Theories (SMT) problem. We postulate four fundamental axioms that bridge the gap between abstract logic and concrete execution.

C.1 Axiom 1: Closed-World Data Manifestation ($\mathcal{C}_{cwa}$)

Formal Definition: We enforce the Closed-World Assumption (CWA) combined with Operation Grounding.

1. Domain Closure: The solver's universe $\mathcal{U}$ is strictly bounded by the entities present in $\mathcal{T}$. We enforce an Injectivity Constraint to prevent entity aliasing:

$$\forall v_i, v_j \in V_{\mathcal{T}}, \quad i \neq j \implies \texttt{Distinct}(\phi(v_i), \phi(v_j)) \tag{8}$$

2. Quantifier Expansion via Grounding: Higher-order aggregation operators (e.g., max, rank) are computationally expensive in pure first-order logic. We strictly ground these operations into finite conjunctions over the table instance $V_{\mathcal{T}}$:

$$\text{Verify}(x = \max(Col)) \iff \bigwedge_{v \in Col \setminus \{x\}} (x \geq v) \tag{9}$$

This axiom ensures that "Superlative" reasonings (e.g., "highest", 'best") are verified by exhaustive comparison against the ground truth data, rather than abstract symbolic derivation.

C.2 Axiom 2: Topological & Type Consistency ($\mathcal{C}_{struct}$)

Formal Definition: We enforce strict typing and preserve the tabular topology. Let $\Sigma = \{\mathbb{R}, \mathbb{Z}, \mathbb{S}\}$ be the sort system.

- **Type Safety:** For any operation $f(x, y)$, operands must satisfy signature constraints (e.g., arithmetic requires $\mathbb{R}/\mathbb{Z}$, string operations require $\mathbb{S}$).
- **Topological Adjacency:** The table implies implicit relational constraints. If row r_i visually precedes r_{i+1}, we assert an ordinal axiom to support sequential reasoning: $\text{Index}(r_{i+1}) = \text{Index}(r_i) + 1$.

C.3 Axiom 3: Proof by Refutation ($\mathcal{C}_{refute}$)

Formal Definition: The verification relies on Reductio ad Absurdum. Let $\mathcal{K}$ be the knowledge base. To verify step z_t, the solver attempts to find a Counter-Example Model $\mathcal{M}$ that satisfies the negation of z_t:

$$\text{Result} = \texttt{CheckSat}(\mathcal{K} \wedge \neg z_t) \tag{10}$$

- **Status UNSAT:** No counter-example exists ($\neg\exists\mathcal{M}$). Thus, z_t is a necessary logical consequence ($\mathcal{K} \models z_t$). $\rightarrow$ Valid.
- **Status SAT:** A model $\mathcal{M}$ is found where $\mathcal{K}$ holds but z_t is false. This $\mathcal{M}$ serves as diagnostic evidence of spurious correlation. $\rightarrow$ Invalid.

C.4 Axiom 4: Monotonic Context Accumulation ($\mathcal{C}_{seq}$)

Formal Definition: Reasoning is a sequential process of *Deduction* and *Definition*. The knowledge base $\mathcal{K}_t$ evolves dynamically:

$$\mathcal{K}_t = \begin{cases} \mathcal{K}_{t-1} \wedge z_{t-1} & \text{if } \texttt{Type}(z_{t-1}) = \text{DEFINITION} \quad \text{(Axiom Injection)} \\ \mathcal{K}_{t-1} \wedge z_{t-1} & \text{if } \mathcal{K}_{t-1} \models z_{t-1} \quad \text{(Validated Deduction)} \end{cases} \tag{11}$$

This formulation acknowledges that the solver's context expands not only through verifying logical consequences but also through accepting explicit user-defined rules (e.g., "Win = 3 points"), provided they do not contradict the prior state $\mathcal{K}_{t-1}$.

D Downstream Applications: From Audit to Alignment

The dense, step-level signals provided by our TrustTable Solver transform the verification framework into a differentiable oracle, facilitating two key downstream applications: *Logic-Aware Inference* and *Process-Supervised Training*.

1. Logic-Weighted Majority Voting (Inference Time). Standard self-consistency methods (Wang et al., 2023) aggregate answers based solely on frequency, failing to filter out *spurious correlations* where the model derives the correct answer via flawed logic (e.g., hallucinated table joins). We propose a reliability-aware voting mechanism. Let $\{(Z_i, A_i)\}_{i=1}^{K}$ be K reasoning paths sampled from the Neural model. We define the validity weight w_i of path i as the strict conjunction of the Solver's axiomatic audits over all T steps:

$$w_i = \prod_{t=1}^{T} \mathbb{I}\left(\Phi(z_{i,t}, \mathcal{T}) \equiv \text{TRUE}\right) \tag{12}$$

where Φ represents the composite constraint system (Axioms 1–4) defined in App. C. The final answer A^* is selected by maximizing the logic-weighted probability:

$$A^* = \arg\max_{a \in \mathcal{A}} \sum_{i=1}^{K} w_i \cdot \mathbb{I}(A_i = a) \tag{13}$$

By enforcing $w_i \in \{0, 1\}$, we perform Hard Logical Pruning, ensuring that the consensus is built exclusively upon trajectories that respect the Closed-World Assumption and Topological Consistency.

2. Process-Supervised Rejection Sampling (Training Time). To internalize the symbolic constraints into the neural model, we employ *Rejection Sampling Fine-Tuning (RFT)*. Unlike standard outcome-based RFT, which filters samples solely based on final answer correctness ($A_{pred} = A_{gold}$), our framework enforces a rigorous Process-Level Filtering protocol. For each query Q, we sample k reasoning paths. A trajectory Z is retained for the fine-tuning dataset $\mathcal{D}_{SFT}$ if and only if it is free of logical fractures:

$$\mathcal{D}_{SFT} = \{(Q, Z) \mid \text{Exec}(Z) = A_{gold} \wedge \bigwedge_{t=1}^{|Z|} \text{Verifier}(z_t) = \textsc{Pass}\} \quad (14)$$

This mechanism specifically targets Type II Errors. By discarding paths that fortuitously yield the correct answer but fail the Z3 audit (e.g., local max fallacy), we construct a high-fidelity corpus that teaches the model to align with the underlying logic of the table schema, rather than merely fitting the answer distribution.

E Theoretical Analysis: Error Disentanglement via Orthogonal Constraints

To provide a rigorous justification for our architecture, we formulate the TableQA inference process within an Information Theoretic framework (Cover and Thomas, 2006). We formalize how TrustTable minimizes the error bound by enforcing the *orthogonal disentanglement* of factual grounding and logical satisfiability, a property absent in monolithic solvers.

Preliminaries. Let $\mathcal{X}$ denote the context $(Q, \mathcal{T})$ and $\mathcal{Y}$ the target answer space. We define the **Intrinsic Entropy** $H^*(\mathcal{Y}|\mathcal{X})$ as the aleatoric uncertainty inherent in the data (e.g., ambiguity), constituting the irreducible Bayes error. The objective is to minimize the *Excess Entropy*: $\Delta H = H_{\text{model}}(Y|\mathcal{X}) - H^*(\mathcal{Y}|\mathcal{X})$.

Proposition 1 (Structural Deficiency of E2E Models). We model inference as a Markov chain $\mathcal{X} \to Z \to Y$, where Z is the latent reasoning path. Standard E2E models approximate the composite mapping via a continuous distribution $P_\theta(Y|\mathcal{X})$. However, recent theoretical works (Merrill and Sabharwal, 2024; Dziri et al., 2023) suggest that Transformer-based models, when approximating discrete symbolic operations (e.g., arithmetic, sorting) via soft attention, suffer from non-vanishing **Computational Friction** (ϵ_{comp}). By the Chain Rule of Entropy, the lower bound is:

$$H_{\text{E2E}}(Y|\mathcal{X}) \geq H^*(\mathcal{Y}|\mathcal{X}) + H_{\text{reason}}(Z|\mathcal{X}) + \epsilon_{\text{comp}} \quad (15)$$

where $\epsilon_{\text{comp}} > 0$ represents the residual entropy arising from the architectural mismatch between continuous representations and discrete logic, which cannot be eliminated solely by scaling (Liu et al., 2024).

Proposition 2 (Deterministic Collapse via Symbolic Decomposition). TrustTable explicates the latent variable Z into executable code. Since the execution engines are deterministic Turing machines, the conditional entropy $H(Y|Z, \mathcal{X})$ vanishes. Invoking the Data Processing Inequality (DPI) (Cover and Thomas, 2006), we shift the verification target from the stochastic outcome Y to the structured program Z:

$$H_{\text{Ours}}(Y|\mathcal{X}) \leq H(Z|\mathcal{X}) + \underbrace{H(Y|Z, \mathcal{X})}_{0} = H(Z|\mathcal{X}) \quad (16)$$

This decomposition structurally eliminates the computational friction ($\epsilon_{\text{comp}} \to 0$). The problem reduces to minimizing the reasoning uncertainty $H(Z|\mathcal{X})$ via external auditing.

Proposition 3 (Optimality of Orthogonal Decoupling). To minimize $H(Z|\mathcal{X})$, we introduce a verification signal $\mathcal{V}$. In monolithic verification (e.g., Self-Check), factual and logical errors are statistically coupled in the model's latent space. TrustTable decomposes $\mathcal{V}$ into two functions acting on disjoint properties of Z, ensuring mathematical orthogonality:

- **Factual Grounding ($\mathcal{V}_{pd}$):** Utilizing **Pandas**, this verifies the consistency of variable instantiation against the schema $\mathcal{T}$. It constraints the *Valuation Space* of Z, ensuring every symbol $s \in Z$ maps to a valid entity in $\mathcal{T}$ (Pasupat and Liang, 2015).
- **Logical Satisfiability ($\mathcal{V}_{z3}$):** Utilizing the **Z3 SMT Solver** (de Moura and Bjørner, 2008),

this verifies the validity of the deduction rules independent of data values. It constraints the *Function Space* of Z, ensuring the derivation ϕ is satisfiable (SAT(ϕ)) regardless of specific instantiations.

Since the validity of a logical form (Syntax) is independent of variable instantiation (Semantics) in First-Order Logic, we postulate conditional independence: $P(\mathcal{V}_{pd}, \mathcal{V}_{z3}|Z) \approx P(\mathcal{V}_{pd}|Z)P(\mathcal{V}_{z3}|Z)$. Consequently, the **Mutual Information** gain is additive:

$$I(Z; \mathcal{V}_{pd}, \mathcal{V}_{z3}) \approx I(Z; \mathcal{V}_{pd}) + I(Z; \mathcal{V}_{z3}) \quad (17)$$

This additivity implies that TrustTable maximizes the reduction of the hypothesis space entropy compared to entangled verifiers, which suffer from information redundancy ($I(Z; \mathcal{V}_{entangled}) < \sum I_k$).

Theorem 1 (Optimal Error Bound). Combining Propositions 2 and 3, the error bound of TrustTable converges strictly lower than that of E2E models:

$$H_{\text{Ours}}(Y|\mathcal{X}) \to H^*(\mathcal{Y}|\mathcal{X}) + \underbrace{\left(H_{\text{prior}}(Z|\mathcal{X}) - \sum_{k \in \{pd, z3\}} I(Z; \mathcal{V}_k) \right)}_{\text{Minimized Reasoning Uncertainty}} \quad (18)$$

Corollary (Structural Convergence). The derivation highlights a distinction in convergence regimes. While E2E models rely on asymptotic convergence (requiring infinite data to marginalize ϵ_{comp}), TrustTable achieves **Structural Convergence**. By enforcing orthogonal constraints on both Valuation and Function spaces, the framework guarantees a valid hypothesis subspace $\mathcal{Z}_{\text{valid}} = \mathcal{Z} \setminus (\mathcal{Z}_{hallucination} \cup \mathcal{Z}_{fallacy})$, theoretically eliminating Type 2 (Spurious) errors independent of sample size.

F Detailed Derivations of Theoretical Analysis

In this section, we provide the complete step-by-step derivations for the Propositions and Theorem presented in Section E. We base our analysis on standard Information Theory definitions, including Entropy $H(\cdot)$, Conditional Entropy $H(\cdot|\cdot)$, and Mutual Information $I(\cdot;\cdot)$.

F.1 Proof of Proposition 1: The Structural Deficiency of E2E Models

Statement. *The entropy lower bound of an E2E model is given by* $H_{E2E}(Y|\mathcal{X}) \geq H^*(\mathcal{Y}|\mathcal{X}) + H_{reason}(Z|\mathcal{X}) + \epsilon_{comp}$.

Proof. Let the inference process be modeled as a Markov chain $\mathcal{X} \to Z \to Y$, where $\mathcal{X}$ is the context, Z is the latent reasoning path, and Y is the final answer. An ideal end-to-end model attempts to learn the distribution $P(Y|\mathcal{X})$ by marginalizing over the latent Z:

$$P(Y|\mathcal{X}) = \sum_{z \in \mathcal{Z}} P(Y|z, \mathcal{X}) P(z|\mathcal{X}) \quad (19)$$

The uncertainty of the output Y is bounded by the joint entropy of the generative process. By the **Chain Rule of Entropy**, we have:

$$H(Y, Z|\mathcal{X}) = H(Z|\mathcal{X}) + H(Y|Z, \mathcal{X}) \quad (20)$$

Step 1: Decomposing the Reasoning Entropy $H(Z|\mathcal{X})$. The term $H(Z|\mathcal{X})$ represents the uncertainty in generating the reasoning path. This can be decomposed into the irreducible data uncertainty (Aleatoric) and the model's epistemic uncertainty:

$$H(Z|\mathcal{X}) = H^*(\mathcal{Y}|\mathcal{X}) + H_{\text{reason}}(Z|\mathcal{X}) \quad (21)$$

where $H^*(\mathcal{Y}|\mathcal{X})$ is the *Intrinsic Entropy* (Bayes Error) due to ambiguity in $\mathcal{X}$, and H_{reason} is the uncertainty in synthesizing the correct symbolic form.

Step 2: Defining Computational Friction ϵ_{comp}. The term $H(Y|Z, \mathcal{X})$ represents the uncertainty of the answer given the reasoning path. For E2E models, the mapping $Z \to Y$ is approximated by neural layers (e.g., Attention mechanisms performing arithmetic). Since continuous functions cannot perfectly represent discrete symbolic operations without infinite precision, there exists a residual entropy:

$$H_{\text{E2E}}(Y|Z, \mathcal{X}) \triangleq \epsilon_{\text{comp}} > 0 \quad (22)$$

This ϵ_{comp} captures the probability mass dispersed over incorrect answers due to arithmetic drift or hallucination, even when the internal reasoning Z is implicitly correct.

Step 3: Lower Bound. Although E2E models output a marginal distribution $P(Y|\mathcal{X})$, the total information required to resolve Y cannot be less than the joint uncertainty of the underlying causal

chain (by the Data Processing Inequality applied to the error propagation). Thus:

$$\begin{aligned} H_{\text{E2E}}(Y|\mathcal{X}) &\geq H(Y, Z|\mathcal{X}) - \delta \\ &\approx H(Z|\mathcal{X}) + H(Y|Z, \mathcal{X}) \\ &= [H^*(\mathcal{Y}|\mathcal{X}) + H_{\text{reason}}(Z|\mathcal{X})] + \epsilon_{\text{comp}} \end{aligned} \tag{23}$$

This concludes the proof. □

F.2 Proof of Proposition 2: Deterministic Collapse via Symbolic Decomposition

Statement. *By utilizing deterministic execution, TrustTable eliminates computational friction, yielding* $H_{Ours}(Y|\mathcal{X}) \leq H(Z|\mathcal{X})$.

Proof. TrustTable explicitly generates a symbolic program Z (e.g., Python code). The final answer is obtained via a deterministic execution engine $\mathcal{E}_{ngine}$ (e.g., Python Interpreter):

$$Y = \mathcal{E}_{ngine}(Z) \tag{24}$$

Since Y is a deterministic function of Z (and potentially $\mathcal{X}$ for context), the conditional probability $P(Y|Z, \mathcal{X})$ collapses to a Dirac delta function. Consequently, the conditional entropy vanishes:

$$H(Y|Z, \mathcal{X}) = 0 \implies \epsilon_{\text{comp}} = 0 \tag{25}$$

We now apply the property that the entropy of a function of a random variable is less than or equal to the entropy of the variable itself ($H(f(X)) \leq H(X)$):

$$H_{\text{Ours}}(Y|\mathcal{X}) = H(\mathcal{E}_{ngine}(Z)|\mathcal{X}) \leq H(Z|\mathcal{X}) \tag{26}$$

This proves that by decoupling execution, the system's error bound is strictly limited by the program synthesis uncertainty $H(Z|\mathcal{X})$, effectively removing the execution error term. □

F.3 Proof of Proposition 3: Additive Gain via Orthogonal Verification

Statement. *Given orthogonal verification signals* $\mathcal{V}_{pd}$ *and* $\mathcal{V}_{z3}$*, the Mutual Information gain is additive:* $I(Z; \mathcal{V}_{pd}, \mathcal{V}_{z3}) \approx I(Z; \mathcal{V}_{pd}) + I(Z; \mathcal{V}_{z3})$.

Proof. We define the verification process as observing a combined signal $\mathcal{V} = (\mathcal{V}_{pd}, \mathcal{V}_{z3})$. The reduction in uncertainty of Z is measured by the joint Mutual Information $I(Z; \mathcal{V}_{pd}, \mathcal{V}_{z3})$. By the **Chain Rule for Mutual Information**:

$$I(Z; \mathcal{V}_{pd}, \mathcal{V}_{z3}) = I(Z; \mathcal{V}_{pd}) + I(Z; \mathcal{V}_{z3}|\mathcal{V}_{pd}) \tag{27}$$

To prove additivity, we must show that $I(Z; \mathcal{V}_{z3}|\mathcal{V}_{pd}) \approx I(Z; \mathcal{V}_{z3})$. This equivalence holds if and only if $\mathcal{V}_{z3}$ and $\mathcal{V}_{pd}$ are statistically independent with respect to the information they provide about Z.

Analysis of Orthogonality: The random variable Z (reasoning chain) contains two distinct types of information features:

- $F_{val}(Z)$: The valuation of variables (data grounding).
- $F_{func}(Z)$: The functional structure (logical validity).

Our verifiers act on disjoint feature spaces:

$$\mathcal{V}_{pd} \leftarrow \text{Check}(F_{val}(Z)), \quad \mathcal{V}_{z3} \leftarrow \text{Check}(F_{func}(Z)) \tag{28}$$

In First-Order Logic and Type Theory, the syntactic validity of a formula (Function Space) is independent of the specific assignment of values (Valuation Space). Thus, observing a grounding error (via Pandas) provides negligible information about the logical satisfiability (via Z3), and vice versa. Mathematically, this implies conditional independence:

$$P(\mathcal{V}_{z3}|\mathcal{V}_{pd}, Z) \approx P(\mathcal{V}_{z3}|Z) \tag{29}$$

Substituting this into the mutual information definition:

$$\begin{aligned} I(Z; \mathcal{V}_{z3}|\mathcal{V}_{pd}) &= H(\mathcal{V}_{z3}|\mathcal{V}_{pd}) - H(\mathcal{V}_{z3}|Z, \mathcal{V}_{pd}) \\ &\approx H(\mathcal{V}_{z3}) - H(\mathcal{V}_{z3}|Z) \quad \text{(due to orthogonality)} \\ &= I(Z; \mathcal{V}_{z3}) \end{aligned} \tag{30}$$

Therefore, the total information gain is the sum of individual gains:

$$I(Z; \mathcal{V}_{total}) \approx I(Z; \mathcal{V}_{pd}) + I(Z; \mathcal{V}_{z3}) \tag{31}$$

This additivity demonstrates that TrustTable minimizes information redundancy, achieving a coverage efficiency that is theoretically superior to entangled verifiers where $I(Z; \mathcal{V}_1, \mathcal{V}_2) \ll \sum I(Z; \mathcal{V}_i)$. □

F.4 Derivation of Theorem 1: Optimal Error Bound

Statement. *TrustTable achieves a strictly lower theoretical error bound defined by the reduced reasoning uncertainty.*

Proof. We start with the result from Proposition 2:

$$H_{\text{Ours}}(Y|\mathcal{X}) \leq H(Z|\mathcal{X}) \tag{32}$$

The term $H(Z|\mathcal{X})$ represents the uncertainty of the generator *before* verification. With the introduction of the verification signals $\mathcal{V} = \{\mathcal{V}_{pd}, \mathcal{V}_{z3}\}$, we aim to minimize the posterior entropy $H(Z|\mathcal{X}, \mathcal{V})$. Using the definition of Mutual Information $H(X|Y) = H(X) - I(X;Y)$, the posterior entropy is:

$$H(Z|\mathcal{X}, \mathcal{V}) = H_{\text{prior}}(Z|\mathcal{X}) - I(Z; \mathcal{V}_{pd}, \mathcal{V}_{z3}) \tag{33}$$

Substituting the additive gain derived in Proposition 3:

$$H(Z|\mathcal{X}, \mathcal{V}) \approx H_{\text{prior}}(Z|\mathcal{X}) - (I(Z; \mathcal{V}_{pd}) + I(Z; \mathcal{V}_{z3})) \tag{34}$$

Finally, considering the intrinsic data entropy $H^*(\mathcal{Y}|\mathcal{X})$, the total error bound for TrustTable converges to:

$$H_{\text{Ours}}(Y|\mathcal{X}) \to H^*(\mathcal{Y}|\mathcal{X}) + \underbrace{\left(H_{\text{reason}}(Z|\mathcal{X}) - \sum_{k \in \{pd, z3\}} I(Z; \mathcal{V}_k) \right)}_{\text{Minimized Reasoning Uncertainty}} \tag{35}$$

Comparing this to the E2E bound (Eq. 24), we observe two strict improvements:

1. The elimination of the strictly positive term ϵ_{comp} (via Prop 2).
2. The maximization of the subtraction term via orthogonal mutual information (via Prop 3).

Thus, $H_{\text{Ours}}(Y|\mathcal{X}) \ll H_{\text{E2E}}(Y|\mathcal{X})$. □

G Error Analysis: What Does TrustTable Miss?

To characterize the remaining leakage of our auditor, we manually audited mis-verified cases sampled from WikiTableQuestions (DeepSeek-chat reproduction) and categorized them into three primary failure modes. Below we cite representative cases from our run, each annotated with its WTQ item ID for reproducibility.

G.1 1. Overly Permissive Fuzzy Grounding (∼48% of Leakage)

To accommodate real-world noisy tables (inconsistent whitespace, partial matches, formatting variations), our FactChecker employs fuzzy string and numeric matching. While this significantly improves recall, it can be too permissive in adversarial cases.

Case 1a — Positional grounding (WTQ `nu-66`). *Question:* "What date is next listed after June 14, 2010?" *Claimed:* December 6, 2010. *CoT step:* "I looked at the cell directly underneath June 14, 2010 in the Season Finale column." *Verifier:* `df['Season Finale'].str.contains("December 6, 2010")` → **PASS**. *Analysis:* The date *exists* in the column, so containment succeeds; the FactChecker cannot verify the *positional* adjacency claim.

Case 1b — Derivation-path grounding (WTQ `nu-145`). *Question:* "Who scored the same number of league goals this season as Gregory Nelson?" *CoT:* "Gregory Nelson has Total= 4 and Europa= 1, so League= 3; Aquaro has Total= 5 and Europa= 2, so League= 3." *Analysis:* The CoT bypasses the table's explicit `League` column in favor of a derived quantity. Our FactChecker validates each atomic numeric claim but does not currently enforce "use the explicit column when available".

G.2 2. Semantic Translation Misalignment (∼40% of Leakage)

A second major source of leakage stems from the natural-language-to-code translation. The Program Synthesizer occasionally generates code that is *syntactically valid and fully executable, yet semantically flawed*.

Case 2a — Unique-count vs occurrence-count (WTQ `nu-67`). *Question:* "How many times was a tournament held in the United States?" *Claimed:* 1. *CoT:* "I list *unique* countries: Japan, US, Canada, Taiwan. . . ; US appears once in this unique list, so 1." *Analysis:* The generated code implements "distinct countries" semantics, whereas the question asks for total occurrences, a wrong aggregation despite valid syntax.

Case 2b — False rejection (WTQ `nu-58`). *Question:* "Who was the top placing competitor?" *CoT claim:* "Ze'ev Friedman" (correct). *ConsistencyMonitor:* `df.sort_values('Placing').iloc[0]['Name']` returns `"Esther Shahamorov"`. *Decision:* **REJECT**, a symmetric false-reject caused by the synthesizer choosing a different but equally valid aggregation.

Table 11: Latency breakdown of TrustTable for a representative 5-step query. Over 98% of the cost is neural; the symbolic solver's contribution is negligible.

Stage	Component	Time
1. Generation	Initial CoT (decoding)	$\sim$1.7s (32%)
2. Decomposition	Step parsing	$\sim$0.9s (18%)
3. Synthesis	Code gen (N LLM calls)	$\sim$2.4s (48%)
4. Execution	Pandas & Z3 solving	$<$0.1s ($<$2%)
Refinement	*single pass (conditional)*	*$\sim$1.9s*

G.3 3. Irrelevant Logical Noise ($\sim$12% of Leakage)

In a minority of cases the CoT exhibits logic that is internally consistent but irrelevant to the question. Since Z3 verifies logical *satisfiability*, not *causal relevance*, it cannot intercept such detours.

Case 3a — Round-trip arithmetic (WTQ `nu-151`).
The CoT rounds $1902 \rightarrow 1900$ and $1998 \rightarrow 2000$, computes gap 100, then subtracts $2{+}2$ to recover 96. Z3 validates $1998{-}1902{=}96$; our auditor does not flag the procedurally odd but mathematically valid detour.

Symmetric False Rejections. For completeness we document two over-correction patterns. Case `FR-1` (WTQ `nu-60`): the table's header label "Date" is read as "Event Date" by the FactChecker code, yielding `COLUMN_NOT_FOUND` despite a valid claim. Case `FR-2` (WTQ `nu-75`): Z3 encodes a universal quantifier not asserted by the CoT, producing a spurious counter-example.

Mitigation roadmap. We plan (1) stricter string thresholds (exact match before substring) for grounding, (2) explicit-column preference when a quantity has both a direct table cell and a derivable form, (3) a lightweight Relevance Check that penalizes reasoning paths whose constraints do not reference any table cell needed for the question, and (4) header-alias tolerance for FR-type issues.

G.4 Latency Breakdown

To clarify the operational overhead, we profile a representative 5-step WTQ query. Table 11 attributes the $\approx$5.0s single-pass latency to each stage.

Two observations follow. First, the symbolic solver contributes less than 2% of wall-clock time; the bottleneck is strictly neural. Second, the N code-synthesis calls are logically independent, so this phase can in principle be parallelized to $O(1)$ latency — a straightforward optimization for future deployments.

G.5 Prevalence of Deterministic Reasoning

A natural question is whether deterministic reasoning (arithmetic, explicit lookup, strict aggregation) covers enough of the TableQA distribution to make a hard-symbolic framework broadly applicable. We address this in two parts.

(1) Prevalence in Existing Benchmarks. Deterministic reasoning is the overwhelmingly dominant paradigm in standard TableQA, though the rate varies by source (Table 12):

- **FinQA (100% Deterministic):** As a numerical reasoning dataset, it relies entirely on objective calculations (e.g., growth rates) and factual extraction.
- **WikiTableQuestions (98.5% Deterministic):** Designed for semantic parsing, it requires rigid compositional logic (e.g., filtering, sorting) akin to SQL execution.
- **SciAtomic (raw, 77.7% Deterministic):** The average across domains is lower — ML 91.1%, Finance 78.4%, Medical 72.7%, Material 68.7%. This stems from its origin in academic papers, where claims frequently mix factual data with subjective evaluations (e.g., *"promising results"* or *"significant improvement"*).

Table 12: Prevalence of deterministic reasoning across source benchmarks and TrustTable-Bench. SciAtomic is further decomposed by subject domain.

Benchmark	Deterministic %
FinQA	100.0%
WikiTableQuestions	98.5%
SciAtomic (raw, avg. over 4 domains)	77.7%
– Machine Learning	91.1%
– Finance	78.4%
– Medical	72.7%
– Material	68.7%
TrustTable-Bench (after filtering)	**100.0%**

(2) TrustTable-Bench Reaches 100% via Strict Filtering. We identified the subjectivity issue in raw SciAtomic and applied a strict filtering protocol during TrustTable-Bench construction to ensure logical rigor. For example, a raw claim *"... the current ratio is 1.70, suggesting a strong ability to*

cover obligations" contains soft logic ("*strong ability*"). Our refinement distills this into a strictly objective fact: *"... the current ratio is 1.70."* Through this refinement, the evaluation in TrustTable-Bench is strictly **100% deterministic**. While soft reasoning exists in raw academic texts, it is virtually absent in high-quality QA benchmarks and has been rigorously excluded from our test set, confirming that TrustTable's Neuro-Symbolic architecture is explicitly tailored to the core, verifiable nature of the TableQA task.

G.6 Isolated Evaluation of the Atomic Decomposer

Because no gold-standard dataset for CoT decomposition in TableQA exists, we conducted an LLM-as-judge evaluation (an independent model grades segmentation and type classification). Table 13 reports the results on 100 randomly sampled WTQ reasoning chains.

Table 13: Evaluation of the Decomposer on N=100 reasoning chains. Decomposer: DeepSeek-chat with enforced JSON mode and role-play prompting. Judge: V3.2 (different model to reduce homogeneity bias).

Dimension	Accuracy
Segmentation Accuracy	92.0%
Type Classification Accuracy	97.0%

The few segmentation errors (8%) were predominantly under-segmentation, where two logical steps were merged into one. Crucially, the downstream FactChecker and Z3Auditor are robust to such deviations: when two steps are merged (e.g., "Revenue is 100 and Cost is 80"), the generated Pandas snippet simultaneously verifies both conditions.

G.7 Z3 Translation Templates

We provide two concrete examples of how a reasoning step is mapped into Z3-executable code. These templates illustrate the NL-to-FOL bridge for auditing.

(1) Arithmetic Calculation. *Reasoning:* "Sales were 50 in 2021 and 60 in 2022, so the total is 110."

```
s = Solver()
sales_21, sales_22, total = Reals('sales_21 sales_22
    total')
s.add(sales_21 == 50, sales_22 == 60)
s.add(total == sales_21 + sales_22)
s.add(total == 110)
print(s.check()) # sat -> accept
```

(2) Logical Comparison / Extrema. *Reasoning:* "Since the margin 15.2 is greater than 12.5, profitability increased."

```
s = Solver()
margin_curr, margin_prior = Reals('margin_curr
    margin_prior')
s.add(margin_curr == 15.2, margin_prior == 12.5)
s.add(margin_curr > margin_prior)
print(s.check()) # sat -> accept
```

By decomposing complex reasoning into atomic algebraic constraints, Z3 deterministically verifies each symbolic step while the LLM handles the natural-language parsing.

H Prompt Templates

To ensure reproducibility and facilitate future research, we illustrate the exact prompts used in our *Neural Generator*, *FactChecker*, *LogicAuditor*, and *Refinement Loop* as individual figures below. All variables enclosed in curly braces (e.g., `{context}`) represent dynamic inputs injected at runtime.

To ensure reproducibility of *TrustTable-Bench*, we display the prompts used for synthesizing Type 1, Type 2, Type 3 and Type 4 samples below.

Use of AI Assistants. We disclose that AI tools (ChatGPT) were used exclusively for linguistic polishing and minor code refactoring to ensure clarity and correctness. All final text and code were audited by the human authors.

Prompt 1: Reasoning Decomposition

System Instruction:
You are a Reasoning Parser for TableQA tasks. Your goal is to break down a raw Chain-of-Thought (CoT) paragraph into atomic, executable steps. For each step, assign a **Type**:

1. **fact**: The step involves looking up specific data, rows, or values in the table.
2. **inference**: The step involves calculation, comparison, logical deduction, or applying rules.

User Input:
```
Raw CoT Text: "{cot_text}"
Task: Decompose this text into atomic steps. Return JSON.
```

Figure 4: **Reasoning Decomposition Prompt.** Used to parse the raw chain-of-thought into atomic executable steps.

Prompt 2: Pandas Code Synthesis

System Instruction:
You are a Python Pandas Expert for TableQA verification. Your goal is to write a Python function `verify_fact(df)` that checks if a natural language claim is supported by the given DataFrame.
ROBUSTNESS RULES (CRITICAL):

1. **Fuzzy String Matching**: Tables often contain hidden spaces. ALWAYS use `.str.strip()` and `.astype(str)`. Use `.str.contains(..., na=False)` for partial matches.
2. **Partial Match**: If a specific time '3:06:02' is mentioned, try to match the most unique entity (like the name) first.
3. **Column Names**: Use exact column names from the Schema.

User Input:
```
Table Schema: {columns}
Sample Data: {sample_data}
Claim to Verify: "{claim}"
```

Figure 5: **FactChecker Prompt.** Instructs the LLM to synthesize robust Pandas code with fuzzy matching capabilities to handle semi-structured table noise.

Prompt 3: Z3 Constraints Synthesis

System Instruction:
You are an expert in Formal Verification. Your task is to verify if a Conclusion follows from the Premise, GIVEN the Table Data context.
STRATEGY: Data-Augmented Verification

1. **The Closed World Assumption**: The provided "Table Context" contains the ACTUAL values from the real world. Treat these values as hard constraints (Axioms).
2. **Handling "Max/Min/Rank"**: If the conclusion claims "X is the highest", DO NOT just look at the premise. Compare X against the list of values provided in the Table Context.
3. **Output**: Write a `solve_logic()` function returning `(bool, model)`.

User Input:
```
Table Context (Ground Truth): {table_context}
Premise: "{premise_text}"
Conclusion: "{conclusion_text}"
Task: Write Python Z3 code to verify the conclusion.
```

Figure 6: **LogicAuditor Prompt.** Enables Generative Auto-formalization by injecting the table context as logical axioms to enforce the Closed World Assumption.

Prompt 4: Logic Refinement

System Instruction:
You are a Formal Logic Auditor. Your role is to evaluate and fortify a reasoning chain that failed a symbolic verifier.
AUDIT PHILOSOPHY:

- **Case A: Logic Leak (Incomplete Proof)**: The reasoning is correct but "leaky" (e.g., saying "7 is max" without proving others are smaller). *Refinement*: Explicitly cite values of ALL competitors to "close the logical world".
- **Case B: Spurious Logic**: Using rules not supported by the table. *Refinement*: Switch to standard lookup.
- **Case C: Hallucination**: Citing non-existent data. *Refinement*: Re-check the table.

User Input:

```
Original Question: "{question}"
Failed Reasoning Trace: "{old_cot}"
Verifier Feedback:
- Faulty Step: "{failed_step}"
- Technical Objection: {reason}
```

Figure 7: **Refinement Prompt.** Guides the model to repair reasoning flaws based on specific symbolic feedback types.

Prompt A: Golden CoT Generation (Type 1)

System Instruction:
You are an expert data analyst. Answer the question based on the table with a HIGHLY DETAILED, NATURAL reasoning process.
Reasoning Style Requirements:

1. **Interpret Implication**: Start by explicitly interpreting what the question implies.
2. **Grounding**: Explicitly mention finding the numbers in the table.
3. **Rationale First**: Explain the 'Why' before the 'How'.
4. **Tone**: Keep the tone natural, step-by-step.

Input:

```
Table: {table_md}
Question: {question}
```

Figure 8: **Type 1 Generation Prompt.** Uses style-conditioned constraints to generate rigorous and faithful reasoning chains.

Prompt B: Adversarial Spurious Generation (Type 2)

System Instruction:
You are a **Confident but Careless Data Analyst**. Your task is to provide a reasoning path that ARRIVES at the user's provided target answer, but relies on **FLAWED logic or data**.
CRITICAL RULES:

1. **NEVER announce your error.** Just state the incorrect value or logic as if it were absolute fact.
2. **Stay Confident.** Maintain a natural, authoritative tone.
3. **Force the Conclusion**: You must force the conclusion to match the 'Target Correct Answer' exactly.

Error Strategy Injection (Dynamic):

- *Arithmetic*: "Perform the calculation with a subtle error (e.g. 60/150 = 0.45) but miraculously arrive at the target answer."
- *Grounding*: "Confidently cite the WRONG data point (e.g. reading from Row 2 instead of Row 1) but still conclude with the target answer."
- *Logic*: "Use a plausible but incorrect formula/definition that coincidentally leads to the target answer."

Input:
```
Question: {question}
Target Answer: {gold_answer}
```

Figure 9: **Type 2 (Spurious) Generation Prompt.** Employs a "Lying Teacher" strategy with dynamic error injection to synthesize adversarial "Right Answer, Wrong Reason" samples.

Prompt C: Fully Wrong Generation (Type 3)

System Instruction:
You are a flawed reasoner. Generate a response where **BOTH the reasoning process and the final answer are INCORRECT**.

1. Misinterpret the question or look at the wrong column.
2. Ensure the final answer is DIFFERENT from the hidden truth.
3. Keep the tone natural and plausible to a careless reader.

Input:
```
Question: {question}
(Hidden Truth: {gold_answer} - DO NOT OUTPUT THIS)
```

Figure 10: **Type 3 Generation Prompt.** Instructs the model to hallucinate both the reasoning and the answer.

Prompt D: Calculation Error Generation (Type 4)

System Instruction:
You are a Data Analyst with **perfect logical reasoning skills but poor calculation abilities**. Your task is to generate a response where:

1. **Logic is Perfect**: You MUST correctly identify the relevant rows, columns, and numbers from the table.
2. **Formula is Correct**: You MUST set up the correct mathematical operation needed to answer the question.
3. **Calculation is Wrong**: When performing the final arithmetic step, you MUST **intentionally make a calculation error** (e.g., decimal error, simple addition/division mistake) to arrive at a WRONG answer.

The final answer must be different from the 'Hidden Truth'.
Input:
`Table: {table_md}`
`Question: {question}`
`(Hidden Truth: {gold_answer} - DO NOT OUTPUT THIS)`
Output Format:
`{"correct_logic_wrong_math_cot": "...", "incorrect_answer": "..."}`

Figure 11: **Type 4 (Calculation Error) Generation Prompt.** Simulates *Computational Friction* (ϵ_{comp}) by enforcing valid grounding and logic but injecting intentional arithmetic noise in the final execution step.

Prompt E: Standard CoT Verifier – Per-Step Judgment

System Instruction:
You are a strict per-step Logic Auditor.
User Prompt:
You are a strict Logic Auditor verifying ONE atomic reasoning step against a Table.

```
### Table
{table}

### Question (for context)
{question}

### Reasoning Step to verify (one atomic claim)
Step type: {step_type}
Step content: {step}

### Task
```

Is this ONE reasoning step correct based on the table?

- For `fact` steps: does the step cite a value that actually exists in the table at the stated location?
- For `inference` steps: is the computation/aggregation correct given the table rows?
- For `logic` steps: is the deduction/comparison valid?

Respond with a short analysis, then end with exactly `STEP_JUDGMENT: ACCEPT` or `STEP_JUDGMENT: REJECT`.

Figure 12: **Standard CoT Verification Prompts.** Our Standard CoT baseline shares TrustTable's decomposer: each atomic step is judged individually by Prompt E. Any STEP_JUDGMENT: REJECT short-circuits the whole chain.

Prompt F: Programmatic Verification (PoT / Code-Based)

System Instruction:
You are a Computational Logic Auditor. Your goal is to verify a "Reasoning Trace" by converting it into an executable Python verification script using Pandas.
INSTRUCTIONS:

1. **Decompose**: First, break down the reasoning trace into atomic verification steps in comments.
2. **Implement**: Write a Python function `def verify_reasoning(df):` that checks each step against the DataFrame `df`.
3. **Assert**:
   - Verify specific data claims (e.g., `assert df.iloc[0]['Year'] == '2005'`).
   - Verify calculations (e.g., `calculated_sum = df['Points'].sum(); assert calculated_sum == 10`).
   - Verify the final answer consistency.
4. **Return**: The function must return `True` ONLY if all checks pass. If any check fails or data is missing, return `False`.

ROBUSTNESS RULES:

- The dataframe `df` contains strings. You MUST convert columns to numeric types (e.g., `pd.to_numeric`) before doing math.
- Handle formatting (e.g., remove ',' or '$') before conversion.
- Use strict assertions.

Output Format:
Return ONLY the python code block containing the function.

```
def verify_reasoning(df):
# Step 1: ...
# Code ...
return True
```

Input Template:

```
Table Schema & Data Snippet: {table_str}
Question: {question}
Candidate Reasoning to Verify: "{reasoning}"
Predicted Answer: "{answer}"
Task: Generate the Python verification code.
```

Figure 13: **PoT Verification Prompt.** Instructs the model to generate an executable Pandas script to verify the factual grounding and arithmetic logic of a reasoning trace.

Prompt H: VeriCoT – FOL Translation per Step

Task: Translate ONE self-contained CoT step into Z3 Python code for FOL verification. You receive paired evidence: `cot_evidence` (what the CoT claims) and `table_evidence` (actual table values). The solver performs TWO checks: (1) Premise consistency: do CoT-claimed values agree with the table? (2) Deductive validity: does the step's conclusion follow from the premises?

Inputs:

```
### Prior verified steps (additional axioms you may use)  {prior_bullets}
### Current step to verify  Step {step_id} ({step_type}): {step_conclusion}
### CoT evidence  {cot_evidence_bullets}
### Table evidence (ACTUAL values)  {table_evidence_bullets}
### Final CoT claim (only for LAST step)  CoT final: {final_conclusion}; Claimed answer: {claimed}
```

Part A — Premise Consistency Check (detects fabricated values):

A1. Declare a Real/Int variable per `var` in `cot_evidence` (e.g., `john_total_cot = Real('john_total_cot')`).

A2. Declare a SECOND variable per `var` in `table_evidence` (e.g., `john_total_tab`).

A3. Assert CoT values: `s.add(john_total_cot == 20)`.

A4. Assert table values: `s.add(john_total_tab == 12)`.

A5. Assert APPROXIMATE equality with tolerance: for numeric vars use `Abs(x_cot - x_tab) <= max(0.01, 0.005*Abs(x_tab))` (0.5% rel. or 0.01 abs.).

A6. `s.check()` **UNSAT** ⇒ fabrication, print `'UNSAT_PREMISE'`; **SAT** ⇒ proceed to Part B.

Part B — Deductive Validity Check:

B1. Declare any additional variables needed for the step's conclusion.

B2. Encode the step's conclusion as an expression.

B3. `s.push(); s.add(Not(conclusion)); r=s.check(); s.pop()`
UNSAT ⇒ conclusion entailed, print `'UNSAT_CONCLUSION'`.
SAT ⇒ counter-example exists, print `'SAT_COUNTER'`.

Rules:

- Use SEPARATE Z3 variable names for cot vs tab (`x_cot`, `x_tab`).
- If `table_evidence` lacks a variable asserted by the CoT, assume `UNSAT_PREMISE` (fabrication).
- Final print must be one of: `UNSAT_PREMISE` / `UNSAT_CONCLUSION` / `SAT_COUNTER` / `SAT_PREMISE_ONLY`.

Figure 14: **VeriCoT.** Our augmented baseline injects table cells as FOL axioms: Part A checks premise consistency against the table, Part B checks deductive validity. This extends the original VeriCoT (Feng et al., 2025) no-table variant by closing the Semantic–Symbolic Gap on factual grounding.